

Tries for
Lexical Analysis

Price \$4.50 U.S.
(Canada \$5.95)

The Users Journaltm

April 1992

Volume 10, Number 4

User Interfaces

- Interactive Menu Selections
- Adding a GUI Later

Also:

- A C++ Money Class
- Multi-Copy Math Functions
- User Report: Zinc Class Library

Columns By:
Rex Jaeschke
Kenji Hino
P. J. Plauger
Ken Pugh
Sydney Weinstein



RS-232³

Three powerful tools for programming asynchronous communications
The Grunt Work done – by Greenleaf

NEW C Version!

Greenleaf CommLib Level 2™ Version 3.2

- NEW SUPPORT - Phar Lap 286/DOS-Extender, Rational DOS/16M.
- Device independent function calls supported by twelve drivers – support intelligent and standard multiport boards and/or FOSSIL drivers.
- XMODEM, YMODEM (Batch), ZMODEM (Batch), Kermit, ASCII file transfers. 1K, G, CRC options.
- Access modems via network using Fresh Technology MODEM Assist™ or CDS Sparkle program.
- XON/XOFF, RTS/CTS and DTR/DSR flow controls.
- Up to 115,200 baud. 8250, 16450 UART, and 16550 FIFO modes. Ring-buffered, all four interrupts serviced. Buffers to 65K. Fast receive interrupt option for tight spots.
- Unlimited number of ports for all popular multi-port boards including DigiBoard, StarGate and Arnet.
- More than forty modem control functions.
- Includes keyboard functions, Ctrl-Break Handler, unique WideTrack™ Receive feature.

NEW C++ Product!

Greenleaf Comm++™

- True C++ Class Library.
- Device Independence through Inheritance.
- Supports Zortech C++, Borland C++ and TopSpeed C++.
- DOS, OS/2, Windows.
- IBM PC, XT, AT, PS/2 and Compatibles.
- COM1..COM4.
- Baud Rates to 115,200.
- XON/XOFF, RTS/CTS Flow Controls.
- XMODEM, Kermit File Transfer Protocols.
- VT52, VT100 subset, ANSI Terminal Emulation plus TTY Emulation.
- Hayes Modem Control Classes.
- Line and Link Level Controls & Status.
- Extensive Examples help you get started.

Debugging Tool!

Greenleaf ViewComm™

- Data Line Monitor - Full featured RS-232 "Datascope" runs on your PC or Laptop.
- Monitor, Capture Data in ASCII, EBCDIC, Baudot to 115,200 baud.
- Capture to buffer or file, Review, String Search.
- Unlimited Triggers on Modem Status, Data Patterns, including wildcards.
- View Data in Hex, Octal, Decimal, Binary, with Graphic Characters for Control Codes.
- Timestamps, Variable Resolution, for each character, down to 1 msec.
- Source Mode lets you send from file, keyboard or both.
- File format & include files provided so you can analyze captured data.
- ViewComm will pay for itself the first time you use it.

All Greenleaf Libraries Come With:

- ✓ Free Source Code, Free Tech Support, Free BBS, Newsletter, Reference Guide
- ✓ Online Documentation System for MS Advisor, Norton, PDQPlus (With engine)
- ✓ Top Rated Documentation
- ✓ Support all major C and C++ compilers for DOS
- ✓ Exclusive **Gold Support Card** support available (Extra charge) guarantees upgrades shipped automatically, plus 800-numbers for Tech Support and 9600 baud BBS.
- ✓ No Royalties, Money-Back Guarantee, of course.

"If you are seriously contemplating writing an application that uses interrupt driven asynchronous routines, you absolutely can't go wrong with CommLib from Greenleaf Software." - Telecomputing Magazine

"Greenleaf CommLib is the smoothest toolkit I own. 'OUTSTANDING' sums it up!" - Larry Dalton

"When developing communications software, there often comes a time when your software just doesn't quite work and it can be devilish to divine just what the problem is. Greenleaf has a simple, inexpensive, and remarkably effective solution. The product is ViewComm." - Boardwatch Magazine

☎ Call today for complete information, demo or to order. MasterCard, VISA, AmEx, approved Purchase Orders.

Greenleaf CommLib v3.2	\$359
Greenleaf Comm++v1.0	\$199
Greenleaf ViewComm v3.0	\$399

1-800-523-9830

214-248-2561 FAX 214-248-7830

BBS (Free Demos) 214-250-3778

Greenleaf Software Inc.

16479 Dallas Parkway, Suite 570
Dallas, TX 75248

✦ Request 240 on Reader Service Card ✦



GREENLEAF

Software®

"I can't believe it's not UNIX."

—Sean Fulton, UNIX Today!

Take it from the critics, Coherent is so close to UNIX, you won't believe your eyes. Or the price.

"Mark Williams Co. seems to have mastered the art of illusion; Coherent comes so fully qualified as a UNIX clone, you find yourself thinking 'I can't believe it's not UNIX.'"

—Sean Fulton, UNIX Today!, November 26, 1990

"...(Coherent) may be the best thing that has happened to UNIX yet."

—William Zachmann, PC Week, November 5, 1990

"If you want to come as close as you can to real UNIX for a low price, COHERENT can't be beat."

—Warren Keuffel, Computer Language Magazine, November 1990

"If you want a UNIX-like development and learning system for less than \$100...I don't see how you can go wrong with Coherent."

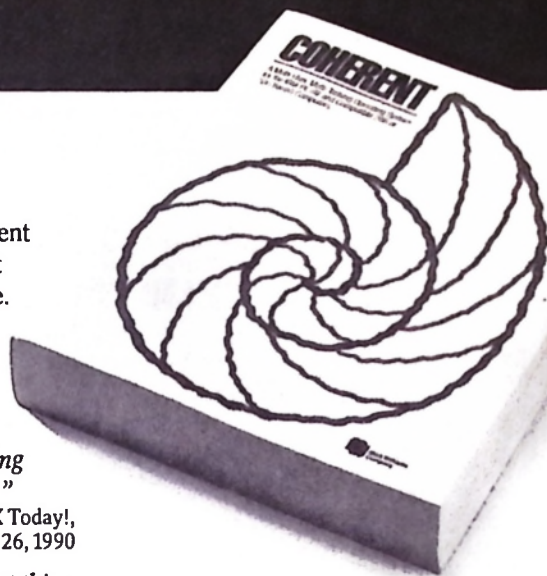
—David Fiedler, BYTE Magazine, November 1990

OVER 30,000 USERS, AND MORE EVERY DAY!

Why is Coherent now the world's best-selling UNIX clone?

	MWC COHERENT Version 3.2	SCO XENIX 286 Version 2.3.2
No. of Manuals	1	8
No. of Disks	5	21
Kernel Size	64K	198K
Install Time	20-30 min.	3-4 hours
Suggested Disk Space	10 meg	30 meg
Min. Memory Required	640K	1-2 meg
Performance*	38.7 sec	100.3 sec
Price	\$99.95	\$1495.00

*Byte Exec benchmark, 1000 iterations on 20 MHz 386.
Hardware requirements: 1.2 meg 5 1/4" or 1.4 meg 3 1/2" floppy, and hard disk.



NEW RELEASE 3.2

\$99.95*

Because like the original UNIX, Coherent is a powerful, multi-user, multi-tasking development system with a complete UNIX-compatible kernel and C compiler.

Features include Lex and Yacc, a vi editor, SCSI support and UUCP capabilities.

And Coherent comes with a full set of over 200 UNIX commands including text processing, program development, administrative and maintenance functions. All of it fully documented in our highly acclaimed 1200 page manual.

WHAT UNIX WAS MEANT TO BE.

Unlike current versions of UNIX, Coherent is lean and efficient. Small and beautifully powerful, the way UNIX was originally designed.

Coherent runs on affordable 286 as well as 386 and 486 based IBM-PCs and compatibles with space to spare. Enough space to keep DOS co-residing on your hard disk.

And it's so fast to install, so fast to learn and just so fast, Coherent leaves UNIX in the dust.

HARD TO BELIEVE? IT KEEPS GETTING BETTER.

Like Coherent, all Mark Williams products are incredible values. Including regular updates with new and enhanced features. Our write-your-own device driver kit. And COHware, contributed software on diskette.

There's also on-going free technical support via telephone. An active user network and a UUCP Bulletin Board System. Plus, with our new 3.2 release:

- A new Korn shell with sophisticated command-line editing.
- Postscript and PCL support for troff adding access to hundreds of new fonts.
- Improved UUCP support.
- International keyboard and character set support.

TAKE 60 DAYS TO CONVINCE YOURSELF

Will you agree with the critics and Coherent's 25,000-plus users?

Try it. And if you don't think Coherent is everything you ever wanted in UNIX, we'll refund your money. No problem. No hassle.

You can't go wrong. So get to a phone, FAX or mailbox now and order Coherent today. At \$99.95, it's unbelievable.

1-800-MARK WMS

(1-800-627-5967 or 1-708-291-6700)

FAX: 1-708-291-6750

60-DAY MONEY BACK GUARANTEE!



**Mark Williams
Company**

60 Revere Drive
Northbrook, IL 60062

*Plus shipping and handling. Coherent is a trademark of Mark Williams Company. UNIX is a trademark of AT&T. XENIX is a trademark of Microsoft.

Distributors: Australia (07) 266-2270, Czechoslovakia 632-62877, Denmark 42-88-72-49, Finland 47-871-201, France (1) 46-72-80-74, Germany (0511) 53-72-95/(030) 313-7015, Norway 211-0950, Singapore 336-0188, Sweden (0) 660-192-90.

C CODE FOR THE PC

source code, of course

NEW!	DIAPrep (embedded SQL to C translator; supports Oracle, Sybase, SQL Server, XDB, Novell XQL, SQLBase, and QI-Lib)	\$1,500
	Virtual Memory Objects (btrees, lists, arrays, and other multiple memory classes in heap, EMM, and swap file; no royalties)	\$750
	Embedded DOS (full-features, real-time, multitasking, 3.31-compatible DOS for embedded system and self-bootable installations)	\$375
	Spell Time (spelling checker for incorporation into text products; no royalty; large dictionary; small and fast)	\$300
NEW!	Imaging Objects (C++ object library for imaging, PCX & TIFF, convolutions, print with dither, diffusion & halftones)	\$290
	ZIP Image Processor & Victor Image Library Version 2.0 (brightness, contrast, merge images, TIFF/GIF/PCX/bin, HP ScanJet support)	\$290
	McJin++ (C++ class library for modeling & simulation; queues, statistical tools, fourier transforms, differential equations; no royalties)	\$255
	TurboJ++ (Release 3.0; HP, PS, dot drivers; CM fonts; LaTeX; MetaFont)	\$250
	Rogue Wave tools.h++ or math.h++ Class Library (extensive docs)	each \$240
NEW!	PxSQL (SQL for Borland's Paradox Engine; ANSI X3.135-1989 SQL-DML standard; network server not required)	\$180
NEW!	Embedded DOS Utility SDK (C source for standard DOS utilities — FDISK, FORMAT, COMMAND.COM and eight more)	\$170
	c.plib (PostScript generation library for C programs; includes complete graphics, font, rotation & paragraph support)	\$170
	WKS Library Version 2.01 (C program interface to Lotus 1-2-3, dBase, Supercalc 4, Quatro, & Clipper)	\$155
	Empty Shell (run a program in an empty DOS shell; written in assembler for speed and small size; C call interface)	\$150
	Minix Operating System (Version 1.5; Unix-like operating system, includes manual; specify 5.25" or 3.5" diskettes)	each \$150
Updated!	Delorie GCC for MS-DOS (Version 1.05; includes C++, assembler, DOS extender, 387 emulation; complete source code and makefiles)	\$150
	Heap Expander (Version 3.0, virtual memory manager using expanded memory, extended memory and disk; XMS, VCPI, LIM)	\$135
	CBTree (B+tree ISAM driver, multiple variable-length keys)	\$135
	TE Editor Developer's Kit (full screen editor, undo command, multiple windows; with word processing: \$195; Windows 3.0: \$200)	\$130
	XMEM (extended and expanded memory manager; written in assembler with C call interface; swaps to disk; small and fast)	\$130
	Booster Toolkit (floppy disk bootstrap routines, DOS file system, light-weight multitasking, windows, fast memory management)	\$120
	WinMem (unlimited global memory handles for Windows 3.0, 4-byte overhead per block rather than 20, virtual memory for 286)	\$110
	CSIM (discrete event simulator library; clocks, chains, future events, multiple servers, queues, reports)	\$100
	PC/IP (CMU/MIT TCP/IP for PCs; Clarkson drivers, NFS server, Bdale mailer, PCRoute/PCBridge, NDIS/ODI drivers, Beholder, more)	\$100
	Heapman (application memory management for Windows 3.0; 64K bytes of heap space; includes memory browser/debugger)	\$100
	EZThieve (Novell Btrieve access with data dictionary and data manager; no royalties)	\$100
NEW!	EZBase (C interface to dBase files; create, read, write & update .DBF and .DBT; includes index support)	\$100
	CDB for DOS & Unix (database toolkit; ISAM, DDL, relational & network, space reuse, concurrent access; fast, no royalty)	\$95
	Kier FinanceLib (interest, conversions, annuities, depreciation, cash flow, bonds, etc.)	\$95
	PowerSTOR (Version 1.2; extended heap space on extended memory, expanded memory, and/or hard disk)	\$95
	Script Interpreter (a command script interpreter for DOS-based systems; C-like script language; lots of features)	\$90
	Visions 1.20 (text window user interface management system; includes mouse support and background processing)	\$80
	VMM (virtual memory manager by Blake McBride, Version 3.6, LRU pager, dynamic swap file, image save/restore)	\$80
	cbase (C database library, sequential & b+-tree random access, buffered block I/O, file-level locking)	\$75
	TeeDraw (PostScript display of labeled hierarchical trees; DOS & Windows screen display included; Moen algorithm)	\$75
	FinanC (large collection of financial function including bond, inventory, stock portfolio, & cash flow)	\$70
	FlexList (doubly-linked lists of arbitrary data with multiple access methods; specify C or C++)	each \$65
	Foundations-1 C++ Class Library (ASCII record I/O, bit arrays, exception handling, B-tree, persistent objects)	\$60
NEW!	LDB (Loose Data Binder; persistent data objects for C++; handles pointers between objects)	\$60
	Kier DateLib (all kinds of date manipulation; translation, validation, formatting, & arithmetic)	\$60
	Coder's Prolog (Version 3.0; inference engine for use with C programs)	\$60
	MEM.WING (global memory manager for Windows, supports standard C memory allocation calls to "wing" your old C code into Windows)	\$55
	Mini IDL (interface generator for complex data; single inheritance, translator & table generator tools, ASCII external form only)	\$50
	CALC (ASCII algebraic expression evaluator, unlimited parenthesis nesting, symbols, 32 built-in functions, easily extended)	\$50
	Backup & Restore Utility by Blake McBride (multiple volumes, file compression & encryption)	\$50
	Floppy TAR (TAR backup and restore on MS-DOS devices; direct access to non-standard devices)	\$50
	SuperGrep (exceptionally fast, revolutionary text searching algorithm; also searches sub-directories)	\$50
	OBJASM (convert .obj files to .asm files; output is MASM compatible)	\$50
	CLIPS Version 5.0 (rule-based expert system generator; advanced manuals available at additional cost)	\$50
	NH Class Library & Book (basic C++ classes & Data Abstraction and Object-Oriented Programming in C++ in softback by Keith Gorten)	\$50
	Editor Pack (15 public domain editors; including microEmacs 3.11, Stevie, Elvis, Moke, mg2a, DTE, Jove, & Origami)	\$50
NEW!	Micro C Compiler (retargetable C compiler with optimizer, libraries, and utilities; lots of docs, very portable, tables for 5 cpu's)	\$50
NEW!	TOUR (beautiful traveling salesman problem solver, finds minimum length paths quickly, includes graphics & plotting programs)	\$40
	Cint (C interpreter)	\$40
	DES Encryption & Decryption (2500 bits/second on 4.77 MHz PC for on-the-fly encryption at 2400 baud; domestic distribution only)	\$40
	RXC & EGREP Version 2.0 (Regular Expression Compiler and Pattern Matching; finite state machine from regular expression)	\$35
	Database Pack (9 databases — simple to complex: isam, bplus, AVL, SDB, ID, gdbm, Requiem, Ingres89, Postgres)	\$35
	Bison & BYACC (YACC worklike parser generators; documentation; no restrictions on use of BYACC output)	\$35
	Object-Oriented Programming in C++ (code from the book by Naba Barkakati)	\$30
	Spell Pack (6 spelling programs, a hyphenator, 2 utility packs and a 60K word list: Ispell, Microsp, Sp, Cspella, Spell, Dawg, Soundex)	\$30
	REGX Plus (Version 2.0, search and replace string manipulation routines based on regular expressions)	\$30
	GNU Awk & Diff for PC (both programs in one package)	\$30
	Big Number Pack (7 arbitrary precision arithmetic packages in C, one in Fortran but free Fortran-to-C converter is included)	\$30
	Crunch Pack (30 file compression & expansion programs; now includes portable ZIP)	\$30
Updated!	UUCP Pack (UUCP for the PC; UUCP Version 1.11Q by Wonderworks and mail/PC Version 2.5 by Stephen C. Trier)	\$25
	PERL for MS-DOS (C, sed, awk, and shell all rolled into one language; includes hardcopy docs)	\$25
Updated!	Td Version 6.1 (Tool Command Language; add shell programming capability to any command line; elegant command line language)	\$25
	FLEX (fast lexical analyzer generator; new, improved LEX; BSD Version 2.3.6 with docs)	\$25
	Livermore Loops in C (the famous Fortran benchmark transliterated to C; includes technical report as PostScript file)	\$20
	XLISP 2.1 (includes Almy improvements)	\$20
	GNU RCS (FSF's version of the Revision Control System; like Unix's SCCS only better; keeps track of software development)	\$20
	Gperf Version 2.5 (GNU's perfect hash table generator; requires DJ gcc; includes executable; specify C or C++)	each \$20
	SDBM (fast, disk-based hash table manager for really large hash tables; clone of Unix ndbm)	\$20
NEW!	PCAL Personal Calendar (generates PostScript calendar for any month or year, lots of options, personalization file for your dates & schedule)	\$20
	Data	
	Moby Thesaurus (25K root words, 1.2M synonyms; requires signed license agreement)	\$350
	Moby Part-of-Speech (200,000 words and phrases described by prioritized part(s)-of-speech)	\$120
	Moby Words (500,000 words & phrases, 9,000 stars, 15,000 names)	\$80
	Moby Shakespeare (plays, sonnets, etc. ... every last word)	\$60
	Dictionary Word List (234,932 words in alphabetical order)	\$60
	Rogel's 1911 Thesaurus	\$40
	U.S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	\$35
NEW!	CIA World Bank II Database (13MB of maps, 5.7M vectors; coastlines, rivers, political boundaries; Africa, Asia, Europe, N. & S. America)	\$35
	The World Digitized (100,000 longitude/latitude of world country boundaries)	\$30
	Lots 'O Words (160,086 German, 178,430 Dutch, 61,843 Norwegian, 60,453 Italian, 138,257 French, 53,142 English)	\$30
Updated!	Text Pack (1990 CIA World Fact Book, Hacker's Jargon File, Acronym List, Koran, Mormon Scriptures)	\$25
	Dan Klein's Dictionaries (53,091 words and phrases in 25 dictionaries)	\$20

The Austin Code Works

11100 Leafwood Lane

Austin, Texas 78750-3587 USA

much more ... ask for catalog

Free surface shipping for cash in advance

For delivery in Texas add 7%

◆ Request 151 on Reader Service Card ◆

Voice: (512) 258-0785

FAX: (512) 258-1342

E-mail: info@acw.com

MasterCard/VISA

Get Inside WINDOWS!



Debug Windows at the systems level!

Soft-ICE/W takes you inside Windows! Debug and explore with power and flexibility not found in any other Windows debugger! **Soft-ICE/W** allows you to debug at the systems or applications level or simply learn the inner workings of Windows.

- Debug VxD's, drivers and interrupt routines at source level
- Debug interactions between DOS T&SR's and Windows Apps
- Debug programs in DOS boxes
- Display valuable system information
(from the total memory occupied by a Windows application, to the complex internal structures of Windows)

Soft-ICE/W uses the 386/486 architecture to provide break point capabilities that normally require external hardware. Nu-Mega, which pioneered this technology with the introduction of its award winning **Soft-ICE** for DOS, now gives Windows programmers the same debugging power... *and still at a software price.*

Own the debugger that combines the best "view" of Windows internals with the most powerful break points of any software debugger.

Soft-ICE/W . . . Only \$386

CodeView for Windows users: see what you're debugging without flash. **CV/1 version 2.0** runs CodeView in a graphics window while viewing your application screen. Runs on any display that supports Windows.

CV/1 . . . Only \$129

Buy any Nu-Mega product and get CV/1 for **only \$69.**

WHAT THE EXPERTS ARE SAYING

"Soft-ICE for Windows is great! It helped me find, in fifteen minutes, a killer bug in a Windows virtual device driver that had eluded two people for several months. I can't see doing Windows development of any kind — whether writing Windows applications, device drivers, or even DOS programs that have to run under Windows — without it. In addition to being great for finding bugs, Soft-ICE for Windows has been essential for my work on a forthcoming book: on Undocumented Windows. Soft-ICE for Windows goes anywhere and does everything, so it's essential for anyone who wants to poke around inside Windows Enhanced mode. DOS programmers will find it a perfect way to learn how the Windows DOS extender and DPMI server work, and how Windows interacts with DOS. Windows Enhanced mode is the hacker's paradise of the 90s, and Soft-ICE for Windows is the tool that every serious Windows or DOS hacker will need. Nu-Mega has done a brilliant job!"

Andrew Schulman

Software Engineer, Phar Lap Software

Editor, Undocumented DOS

Coauthor, Undocumented Windows (forthcoming)

Call: **(603) 889-2386**

FAX: **(603) 889-1135**

P.O. Box 7780

Nashua, NH 03060-7780 U.S.A.



Nu-Mega
TECHNOLOGIES INC

RISK = NULL

30 DAY

MONEY-BACK GUARANTEE

MICROSOFT WINDOWS IS A REGISTERED TRADEMARK OF MICROSOFT CORP. Soft-ICE/W AND CV/1 ARE TRADEMARKS OF NU-MEGA TECHNOLOGIES, INC.

◆ Request 141 on Reader Service Card ◆

The Users Journal[™]

April 1992
Vol. 10, No. 4

USER INTERFACES

- Porting Command Line User Interfaces to GUIs** *By William Smith* 32
Sometimes it pays to do the user interface *last*.
- A Versatile Menu Program for Turbo C** *By Roger T. Stevens* 41
This system allows users to supply interactive input when selecting an option.

OTHER FEATURES

- Multiple Copy Math Functions** *By Dr. Timothy Prince* 21
Vector chunk math functions exploit parallelism to maximize potential.
- Yet Another C++ Money Class** *By Adolfo DiMare* 58
Exploit the power of advanced hardware by writing functions that produce *multiple* results.
- Lexical Analysis Using Search Tries** *By John W. M. Stevens* 67
This special type of tree is a good fit for language translation applications.

COLUMNS

- Standard C** *P. J. Plauger* 8
- Doctor C's Pointers** *Rex Jaeschke* 89
- On The Networks** *Sydney Weinstein* 99
- Questions & Answers** *Ken Pugh* 105
- CUG New Releases** *Kenji Hino* 114

USER REPORT

- Zinc Interface Library**
- Comments by David Brumbaugh* 115

BOOK REVIEW

- C Express: 250+ Ready-To-Run Assembly Language Routines for Turbo C, Microsoft C, and QuickC**
- Reviewed by Stephen Patten* 95

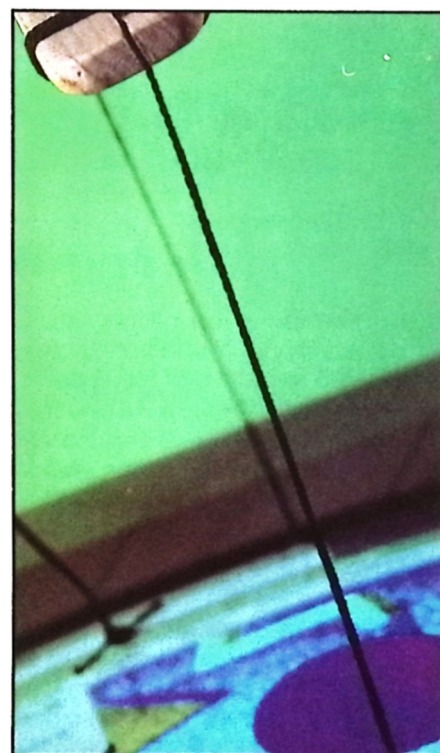


Photo by Robert Ward.

DEPARTMENTS

- Editor's Forum** 6
- Order Forms** 79
- Calendar of Events** 104
- Call For Papers** 107
- Advertiser Index** 112
- New Products** 118
- We Have Mail** 122
- Programmer's Market** 123

WE'VE CHARTED THE WAY WITH WIN++

...for Windows applications
in Borland C++

We've chosen the direction and plotted the course to easy and powerful Windows applications development!

Now with over 150 classes, Win++™ provides a simplified, flexible and powerful high-level C++ interface to make your Windows development efforts a breeze. Classes for setting internal data structures and integrating data into applications are also included.

Put the wind at your back...

Central to Win++ is that it provides a class for each type of display object in Windows. Win++ includes ready-made classes for representing dialog boxes, child controls, pens, brushes, menus, bitmaps, cursors, icons, a printer, a clipboard, and many others. **New classes support the dynamic data exchange management library (DDEML), including clients, servers, advise loops and multiple simultaneous conversations.** Win++ also supports the construction of dynamic-link libraries (DLLs). Use Win++'s ready-made DLLs in your programs and save valuable time, memory and disk space. And, you may distribute Win++'s DLLs with your applications.

It's smooth sailing from here out...

Win++ comes with a comprehensive reference manual and tutorial. Complete source code is included. You can rely on Blaise Computing to provide and support quality add-on libraries. We've been doing so since 1982 — it's our business — our only business.

Success is in the offing...

Win++ requires Windows 3.0 or later and Borland C++. Win++ costs just \$289. We're so convinced that you'll find Win++ essential to your applications that if during the first 60 days you're not completely satisfied, we'll refund your money. No questions asked.

Call our order department toll free at (800) 333-8087! Or, by fax, (510) 540-1938.
Ask about training and consulting services!


BLAISE COMPUTING INC.
819 Bancroft Way Berkeley, California 94710 (510) 540-5441

◆ Request 102 on Reader Service Card ◆

Trademarks are property of their respective holders.

EDITORIAL

Publisher	Robert Ward
Assistant to Publisher	Donna Stucky Ward
Senior Editor	P. J. Plauser
Managing Editor	Diane Thomas
Associate Editor	Martha Masinton
Contributing Editors	Kenji Hino
	Rex Jaeschke
	Kenneth Pugh
	Dan Saks
	Sydney Weinstein
	Leor Zolman

ADVERTISING AND MARKETING

Marketing Director	Jeff Dickey-Chasins
Acct. Manager, East	Ed Day
Acct. Manager, Midwest	Donna Stucky Ward
Acct. Manager, West	Edwin Rothrock
Sales Services	Paula Cobb
Direct Marketing	Bill Uhler

PRODUCTION

Art Director	Susan Schuette Buchanan
Graphic Artist	Twyla Watson Bogaard
Typographer	Ann Bröcker
Production Coordinator	Liza Behmyer

CIRCULATION

Customer Service	Chrissy Trybom
Order Fulfillment	Jodi Leonard

STAFF

Business Manager	Cherilyn Merrill
Technical	Kenji Hino
	Leor Zolman

The C Users Journal is the successor to the C Users' Group Newsletter and The C Journal. Subscribers are automatically enrolled as members of The C Users Group. The C Users Journal and The C Users' Group are services of R&D Publications, Inc., Lawrence, KS.

Entire contents Copyright ©1992 R&D Publications, Inc. No portion of this publication may be reproduced, stored, or transmitted in any form, including computer retrieval, without written permission from the publisher. All Rights Reserved. Printed in the United States of America. Quantity reprints of selected articles may be ordered. By-lined articles express the opinion of the author and are not necessarily the opinion of the publisher.



Advertising: For rate cards or other information on placing advertising in The C Users Journal, contact the advertising department at (913) 841-1631, or write The C Users Journal, 1601 W. 23rd St., Ste. 200, P.O. Box 3127, Lawrence, KS 66046-0127.

Customer Service: For subscription orders and address changes, contact The C Users Journal, 1601 W. 23rd St., Ste. 200, P.O. Box 3127, Lawrence, KS 66046-0127. Telephone (913) 841-1631; FAX (913) 841-2624.

Library Submissions Sought: The C Users' Group Library is the world's largest holding of public domain and Shareware C software and is composed primarily of submissions by members. C programmers are encouraged to submit software to the Library. Send software on disk (no subdirectories on MS-DOS disks please) to The C Users' Group, 1601 W. 23rd St., Ste. 200, P.O. Box 3127, Lawrence, KS 66046-0127. (913) 841-1631.

Trademarks: The C Users Journal, R&D Publications, Inc. Doctor C's Pointers, Rex Jaeschke. UNIX, AT&T Bell Laboratories. XENIX, MS-DOS, OS/2, Microsoft C and QuickC, Windows, Microsoft Corporation. IBM, IBM-PC, Micro Channel, PS/2, International Business Machines Corp. Turbo C, Turbo C++, Borland International. Macintosh, Apple Computer, Inc. Zortech C++, Zortech. Roland MPU401. VAX-VMS, Digital Equipment Corp.

Editor's Forum



I have been thinking a lot about standards lately. That should be no surprise, since I *write* about standards a lot. Still, my perspective has been shifting lately.

When I got involved in the C standard back in 1983, the world was different. Most of us compiler vendors cared intensely about forming a good standard. Many of our potential customers had a different focus. They were more concerned that we didn't mess up a language they had come to depend on. Experience with other major programming language standards was not encouraging. Delays, excess invention, even litigation seemed to be par for the course.

C helped usher in a new era. Here was a standard being formed by active representatives from a broad community. No single vendor dominated the proceedings, nor did a small clique of language inventors. Here was a standard with widespread commercial support from the outset.

And that's what has become the norm. Computing is a big and very competitive business. Complying with standards is of interest to more than just government contractors. The world can't afford bad standards any more. So everyone seems to have something to say about programming standards now.

Even more interesting, programming standards have become a serious matter in the international community. With the growth in international trade, everyone wants a level playing field. Having serious international standards helps the small players compete with the larger ones. It also helps the larger players meet the needs of the smaller markets.

Both ANSI and ISO are scrabbling to adapt to this new state of affairs. The volume of commentary and the pressure to develop timely standards keeps growing, even as the standards become more complex. I don't see how they are going to change fast enough, but I know that they must.

P.J. Plauser
pjp@plauser.uunet

P.J. Plauser
pjp@plauser.uunet.uu.net



Kills bugs dead. Keeps bugs away.

Slash Development Time.

This year and next, C programmers are going to spend millions of hours looking for memory allocation errors. What a shame. Tens of thousands of product dates will slip, and hundreds of thousands of aspirin will be taken. Finally, there's a solution that doesn't involve changing a single line of code.

Meet Your New Watchdog.

MemCheck is an error detection and prevention tool that helps C developers reduce development time and meet application reliability standards. MemCheck will find memory leakage, buffer overwrites, and other nasties with reports that identify the exact location of the culprit. Quickly. Automatically. *Painlessly.*

Greater Certainty.

Whether you're bitten by bugs or just want to make sure none are "hiding," MemCheck is the easiest addition you can make to your software development environment. MemCheck integrates seamlessly with existing C code, with versions that are custom-tailored for all major compilers. Just plug and play — no hassles, no rough edges.

Industrial Strength.

Up your Quality Assurance standards. Find your toughest C bugs. Join the ranks of the major developers who have discovered MemCheck. No product adds so much value with so little effort. Find out why MemCheck is setting standards in software reliability.



≡ **MEMCHECK™**

EFFORTLESS ERROR DETECTION AND PREVENTION

To order, call (313) 996-2944

Suggested retail prices: Microsoft C \$139.95, Borland C \$139.95, Intel Code Builder \$179.95, Watcom C \$179.95, Windows SDK \$179.95
All major credit cards accepted. Fax orders welcome: (313) 747-8519. Please specify compiler and diskette size.

STRATOSWARE CORPORATION, Suite 1500, 1756 Plymouth Road, Ann Arbor, MI 48105

Trademarks mentioned are the property of their respective owners.

◆ Request 414 on Reader Service Card ◆

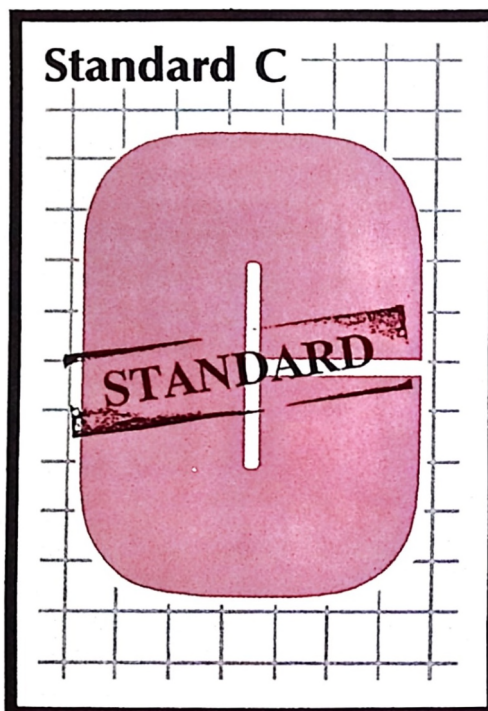
The Header `<stdlib.h>`

Introduction

The header `<stdlib.h>` is a hodgepodge. Committee X3J11 invented this header as a place to define macros and declare functions that had no other sensible home:

- Many existing functions, such as *abs* and *malloc*, had no traditional headers to declare them. X3J11 felt strongly that *every* functions should be declared in a standard header. If such a function seemed out of place in all other headers, it ended up declared in `<stdlib.h>`.
- New groups of macros and functions ended up in new standard headers wherever possible. `<float.h>` and `<locale.h>` are clear examples. Additions to existing groups ended up in existing headers. *strcoll*, declared in `<string.h>`, and *strftime*, declared in `<time.h>`, are also fairly clear. Other macros and functions are harder to categorize. These ended up defined or declared in `<stdlib.h>`.

This header is not the only hodgepodge. I discuss the evolution of the header `<stddef.h>` in *Standard C*, *CUJ* December 1991.



To provide some structure, I organize the functions into six groups:

- integer math (*abs*, *div*, *labs*, and *ldiv*) — performing simple integer arithmetic
- algorithms (*bsearch*, *qsort*, *rand*, and *srand*) — capturing operations complex and widespread enough to warrant packaging as library functions
- text conversions (*atof*, *atoi*, *atol*, *strtod*, *strtol*, and *strtoul*) — determining encoded arithmetic values from text representations
- multibyte conversions (*mblen*, *mbstowcs*, *mbtowc*, *wcstombs*, and *wctomb*) — mapping between multi-byte and wide-character encodings
- storage allocation (*calloc*, *free*, *malloc*, and *realloc*) — managing a heap of data objects
- environmental interactions (*abort*, *atexit*, *exit*, *getenv*, and *system*) — interfacing between the program and the execution environment

I discuss separately how to implement the functions in each of these groups. This month, I cover only the first two groups. I won't bother to present the header as a whole. It's pretty straightforward.

P.J. Plauger is senior editor of The C Users Journal. He is secretary of the ANSI C standards committee, X3J11, and convenor of the ISO C standards committee, WG14. His latest book is The Standard C Library, published by Prentice-Hall. You can reach him care of The C Users Journal or via Internet at pjp@plauger.uunet.uu.net.

Raima Database Engine Captures Fortune 500 With Record Speed



Now Raima Data Manager™
Formerly db_VISTA III

Accelerated Database Performance

Compared to conventional relational databases, retrieval of records can be 10—20—even 50 times faster with Raima Data Manager from Raima Corporation.

Propelling The Biggest Names In Business

Companies like General Motors, Hewlett-Packard, IBM, Eastman Kodak, Rockwell and others are using Raima Data Manager in their competitive environments.

Today's most critical, most demanding applications demand the high performance of Raima Data Manager.

Powerfully Efficient Leading-Edge Technology

Raima's combined technology merges the flexibility of relational databases with the lightning speed and efficient

Raima Data Manager™

The High Performance DBMS

Specifications

Relational B-tree indexing. Network data model. Relational SQL query and report writer. Single & multi-user. Automatic recovery. Built-in referential integrity. Supports: VMS, QNX, ULTRIX, UNIX System V, Berkeley 4.2, AIX, SunOS, SCO, MS DOS, MS Windows, and OS/2. Most C Compilers and LANs supported.

Raima Corporation 3245 146th Place S.E., Bellevue, WA 98007 USA (206)747-5570 Fax: (206)747-1991

International Distributors: Australia: 61 2 419 7177 Belgium: 32 2 734 9818 Finland: 358 080 405350 France: 33 1 46 09 27 84 Germany: 49 7022 34077; 49 214 91051 Italy: 39 49 829 1285
Japan: 81 33 865 2140 Mexico: 52 83 49 53 00 The Netherlands: 31 2159 46814 Norway: 47 2 38 48 88 Singapore: 65 334 0061 Sweden: 46 13 111 588 Switzerland: 41 64 517475
Taiwan: 886 2 552 3277 United Kingdom: 44 992 500919 Copyright ©1992 Raima Corporation. All rights reserved. Please Dale LaFollette

◆ Request 296 on Reader Service Card ◆

storage of the network model. With the program written entirely in C, you can "fine-tune" the Raima Data Manager engine for optimum performance in any application.

Put Yourself In Fast Company

Give yourself the competitive edge of Raima Data Manager:

- Speed—faster access to data
- Portability—supports most environments
- Royalty-free—increase your profits
- Source code availability—total programming flexibility
- Full Raima support services—including training

Whether you're writing a stand-alone DOS application, or one for UNIX accessing thousands of records, Raima Data Manager will put your application on the fast track. Race to the phone and call for more information!

In the U.S. or Canada, call: 1-800-DB-RAIMA

In Washington state or international, call: (206)747-5570

Listing 1 (abs.c)

```
/* abs function */
#include <stdlib.h>

int (abs)(int i)
{ /* compute absolute value of int argument */
  return ((i < 0) ? -i : i);
}

/* End of File */
```

Listing 2 (div.c)

```
/* div function */
#include <stdlib.h>

div_t (div)(int numer, int denom)
{ /* compute int quotient and remainder */
  div_t val;

  val.quot = numer / denom;
  val.rem = numer - denom * val.quot;
  if (val.quot < 0 && 0 < val.rem)
  { /* fix remainder with wrong sign */
    val.quot += 1;
    val.rem -= denom;
  }
  return (val);
}

/* End of File */
```

Listing 3 (labs.c)

```
/* labs function */
#include <stdlib.h>

long (labs)(long i)
{ /* compute absolute value of long argument */
  return ((i < 0) ? -i : i);
}

/* End of File */
```

Listing 4 (ldiv.c)


```
/* ldiv function */
#include <stdlib.h>


ldiv_t (ldiv)(long numer, long denom)
{ /* compute long quotient and remainder */
  ldiv_t val;


  val.quot = numer / denom;
  val.rem = numer - denom * val.quot;
  if (val.quot < 0 && 0 < val.rem)
  { /* fix remainder with wrong sign */
    val.quot += 1;
    val.rem -= denom;
  }
  return (val);
}



/* End of File */
```

Are C Making your program sick?

Program hanging? Memory overwrites?
Operating System overwrites?
Corrupted heap? Memory leaks?
Dangling pointers? Open file pointers?
Invalid calls to free()? 



If your code has a bug that's making you sick, call the doctor — Dr. MD, the run-time memory debugger for C! Sure, you could find it yourself, but how much would it cost you in time, money, and aggravation? 

Dr. MD is the easy and cost effective way to cure your memory corruption ills. Dr. MD gives a complete diagnosis including the source file and line number of your bug. 

 Includes source code. Non obtrusive. 
Easy to customize. Small library.
Hardware independent (386 not required).
Operating System independent (DOS, UNIX, OS/2...).

Order your copy today! Only \$129.

Dr. MD™

 **PCX** 4874 Alberson Ct, Suite 110
San Diego, CA 92130
(619) 259-9797 FAX (619) 481-6474 

◆ Request 130 on Reader Service Card ◆

The C Standard on Integer Math

7.10.6 Integer arithmetic functions

7.10.6.1 The *abs* function

Synopsis

```
#include <stdlib.h>
int abs(int j);
```

Description

The *abs* function computes the absolute value of an integer *j*. If the result cannot be represented, the behavior is undefined.¹³⁰ [FN130. The absolute value of the most negative number cannot be represented in two's complement.]

Returns

The *abs* function returns the absolute value.

7.10.6.2 The *div* function

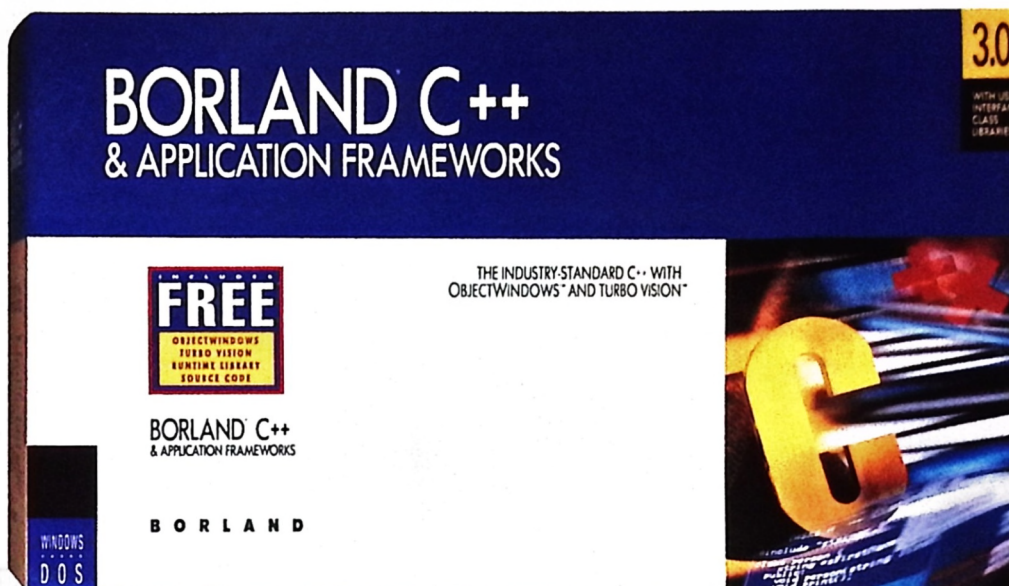
Synopsis

```
#include <stdlib.h>
div_t div(int numer, int denom);
```

Description

The *div* function computes the quotient and remainder of the division of the numerator *numer* by the denominator *denom*. If the division is inexact, the resulting quotient is the integer of lesser magnitude that is the nearest to the algebraic quotient. If the result cannot be represented, the behavior is undefined; otherwise, *quot* * *denom* + *rem* shall equal *numer*.

**NEW!
VERSION 3.0**



...If you program for a living

Borland® C++ & Application Frameworks 3.0 is *the* choice of professional C and C++ programmers for Windows and DOS application development. With unmatched optimizations, powerful tools, unsurpassed Windows development environment and object-oriented application frameworks, Borland C++ & Application Frameworks 3.0 has no equal. Quite simply, if you program for a living, this is everything you need.

OOP, to simplify your life

Borland C++ & Application Frameworks 3.0 simplifies programming by giving you ready-made user interface objects that plug directly into your application. Automatically inherit windows, menus, scroll bars, mouse support and more. Add an editor in just one line. With Object-Oriented Programming (OOP), you get amazing code reusability, extensibility and easier maintenance because applications are built on a base of tested, reliable code.

New features give you incredible programming options!

Just look at some of the enhanced features in Borland C++ 3.0:

- ANSI C and C++ 2.1 and templates
- Global optimizer includes:
 - Global register allocation
 - Local and global common sub-expressions
 - Induction variables
 - Loop and jump optimization
 - Register parameter passing
 - And ten other state-of-the-art optimizations
- Increased C++ compile speed
- Windows and DOS Integrated Development Environments
- Visual ObjectBrowser™ to view class relationships at a glance
- DPMS support for compiler and IDE environments gives you huge capacity
- EasyWin™ library makes it easy to convert your DOS programs to Windows
- Resource Workshop for creating Windows user interfaces visually

- Extensive Microsoft® C compatibility
 - WinSight™ message tracking utility
 - Turbo Debugger® for DOS and Windows
 - Turbo Profiler™ for DOS and Windows
 - Object-oriented Turbo Assembler®
- And with new Borland C++ & Application Frameworks 3.0 you get all of this, plus:
- ObjectWindows™—the application framework for Windows
 - Turbo Vision™—the application framework for DOS
 - Source code for runtime library and application frameworks

Optimized for professionals

Borland C++ 3.0 (\$495^{ms*}) or Borland C++ & Application Frameworks 3.0 (\$749^{ms*}) are optimized for your life-style. But don't wait. Because when it comes to professional programming, there's no better way to earn a living than with Borland C++.

**See your dealer today
or call now
1-800-331-0877, Dept. 5305**

B O R L A N D

The Leader in Object-Oriented Programming

CODE: 5309

*Suggested retail price. All prices are in U.S. dollars. Dealer prices may vary. Copyright © 1991 Borland International, Inc. All rights reserved. All Borland products are trademarks of Borland International, Inc. B1 1451

◆ Request 465 on Reader Service Card ◆

Returns

The *div* function returns a structure of type *div_t*, comprising both the quotient and the remainder. The structure shall contain the following members, in either order:

```
int quot; /* quotient */
int rem; /* remainder */
```

7.10.6.3 The labs function

Synopsis

```
#include <stdlib.h>
long int labs(long int j);
```

Description

The *labs* function is similar to the *abs* function, except that the argument and the returned value each have type *long int*.

7.10.6.4 The ldiv function

Synopsis

```
#include <stdlib.h>
ldiv_t ldiv(long int numer, long int denom);
```

Description

The *ldiv* function is similar to the *div* function, except that the arguments and the members of the returned structure (which has type *ldiv_t*) all have type *long int*.

Using the Integer Arithmetic Functions

Here is a brief summary of the four arithmetic functions:

abs — Call *abs(x)* instead of writing the idiom $x < 0 ? -x : x$. A growing number of Standard C translators generate inline code for *abs* that is smaller and faster than the idiom. In addition, you avoid the occasional surprise when you inadvertently evaluate twice an expression with side effects. Note that on a two's-complement machine, *abs* can generate an overflow.

div — You call *div* for one of two reasons:

- *div* always computes a quotient that truncates toward zero, along with the corresponding remainder, regardless of how the operators / and % behave in a given implementation. This can be important when one of the operands is negative. The expression $(-3)/2$ can yield either -2 or -1, while *div*(-3, 2).quot always yields -1. Similarly, $(-3)\%2$ can yield either 1 or -1, while *div*(-3, 2).rem always yields -1.
- *div* computes both the quotient and remainder at the same time. That can be handy when you need both results. It might even be more efficient if the function expands to inline code that contains only a single divide.

Note that the members of the resulting structure type *div_t* can occur in either order. Don't make any assumptions about the representation of this structure.

labs — is the *long* version of *abs*.


ldiv — is the *long* version of *div*.

Implementing the Arithmetic Functions

Listing 1 shows the file *abs.c*. The absolute value function *abs* is the simplest of the integer math functions. You cannot provide a masking macro, however, because you have to access the value of the argument twice. Some computer architectures have special instructions for computing the absolute value.

graphics-MENU™ 4.0

GUI Toolkit

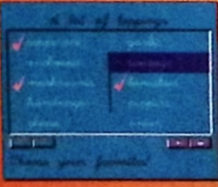


graphics-MENU is a collection of libraries, CASE tools and utilities to enable the software developer to quickly create a professional Graphical User Interface for any application. Extensive flexibility allows modification to color, size and format of most items. Library source is available to allow total control.

- A graphics-MENU interface can contain windows, icons, menus, checked and scrolled lists of items, buttons, text edit window, data entry form menus & more.
- CASE tools are included to provide an on-screen WYSIWYG design environment with


- complete automatic code generation for the target compiler.
- Background tasking capability.
- Memory utilization is conserved by using TPU's for Pascal and many small .OBJ files for C. Also supports EMS (save to disk, Genus version only).
- Complete support for

- both mouse and keyboard.
- Choose native graphics variant to work standalone with your Borland or Microsoft compiler, or external graphics variant to work with MetaWindow or Genus GX graphics.




New features to graphics-MENU 4.0 include a proportionally spaced editor with word-wrap, multi-column checklist menus with stylized check marks and an enhanced file chooser menu.

\$249 graphics-MENU & Data Entry
 \$499 Pro-Pak (full source & lib's)
 \$799 Super Pro-Pak (supports all graphics libs)
 Specify variant:
 • Native graphics (BGI & Microsoft Graphics)
 • External graphics (Genus GX Graphics & META)
 Languages: Turbo Pascal 6.0, TurboC++, Borland C++,
 Microsoft C 6.0, Microsoft C/C++ 7.0



Island Systems
 7 Mountain Road
 Burlington MA 01803
 (617) 273-0421 FAX (617) 270-4437

DEMO AVAILABLE

VISA and MASTERCARD accepted

"The Best Memory Manager"

It's an easy claim to make, but we have the facts to back it up:

Our new QEMM-386 version 6 is the best way to get the most out of memory. It 'pools' all your memory so that it's available in whatever form your programs need—expanded or extended. You don't even need to know the difference. QEMM does it all for you. Instantly. Whereas DOS 5, for example, requires you to figure out what you need, then manually allocate memory and re-boot every time you need to change.

"It's nothing less than a dream come true"
—Steve Gibson InfoWorld 8/26/91

As for the all-important 'conventional' memory area, our new version 6 increases the amount of memory freed-up. Our exclusive 'optimize' feature automatically seeks out TSRs and device drivers and moves them into high memory—the area between 640K and 1 megabyte. All you need do is type 'OPTIMIZE'.

PC Week Ratings	All Charge 386 3.2	Netrom 2.10	QMAPS 2.0	386Max/BlueMax 6.00	QEMM 6.01 Analyst's Choice	Memory Commander 2.1
Software Compatibility	●	●	●	●	●	●
Hardware Compatibility	●	●	●	●	●	●
Reliability	●	●	●	●	●	●
Ease of Use	●	●	●	●	●	●
Memory Management Flexibility	●	●	●	●	●	●
Quality of Documentation	●	●	●	●	●	●

PC Week rated QEMM 6 the best memory manager.



Our latest awards.

QEMM-386 v6 finds more high memory than any other memory manager. Byte Magazine's tests showed it produced net memory gains of 21K to 132K over DOS 5.0 alone, for instance.

Stealth takes you to network and TSR heaven.

Our breakthrough 'Stealth' technology makes available areas normally taken up by ROM. Areas that QEMM-386 can use to load memory-hogging drivers and TSRs. Big programs can get the memory they need to run fast and efficiently. And you get to have your TSRs.

Not every PC can benefit from Stealth. But every PC can benefit from



'Squeeze'—our new feature to accommodate those TSRs that need more memory at start up and less when they're resident. Memory allocation is temporarily increased, then squeezed down after it's needed.

QEMM can use idle video memory to produce a further 96K gain on EGA and VGA systems when running character-based programs.

A priceless \$60 bonus.

QEMM comes with Quarterdeck Manifest, the award-winning analysis program that shows what's going on 'under the hood' of your PC.

Manifest does for memory what PC Tools Deluxe does for disks.

Big Benefits for Windows users, too.

Whether you're running DOS 3, 4, 5, or Windows, QEMM can improve your 386/486's performance.

That means you may not need a faster CPU. You may not even need more RAM. QEMM makes your favorite programs work better by giving them more memory.

QEMM helps you get the most out of the software you own today.

#1

QEMM is not only the best selling memory manager, it's the number one selling PC utility.



Quarterdeck

Quarterdeck Office Systems, 150 Pico Boulevard, Santa Monica, CA 90405 (310) 392-9851 Fax (310) 314-4219
Quarterdeck International Ltd., B.I.M. House, Crofton Terrace, Dun Laoghaire, Co. Dublin, Ireland Tel. (353) (1) 288-1444 Fax: (353) (1) 284-4380

©1992 Quarterdeck Office Systems. PC Week Analyst's Choice Logo. ©1991, Ziff Communications Company. PC Week is a registered trademark and the PC Week Analyst's choice logo is a trademark of the Ziff-Davis Publishing Company. Other trademarks are property of their respective owners. Key to comparison table: Software compatibility is defined as the product's ability to run without problems with the operating systems PC Week Labs used in testing: MS-DOS 3.0, IBM PC DOS 5.0 and Microsoft Windows 3.0, and with a variety of application software, including Quarterdeck's DESQview 2.40, Xerox Ventura Publisher 3.0 for DOS, WordPerfect 5.1 and Arisoft's LANtastic 4.0. Hardware compatibility is defined as the program's ability to run on PC Week Lab's test hardware platforms, which included a 33MHz 486-based PC and a 25MHz 386-based IBM PS/2 Model 70. Reliability ratings are based on the successful and dependable performance of each operating system and application software package under each memory manager. Ease of use is based on ease of installing and configuring each program, based on support for automatic loading features and the intuitive nature of the interface. Flexibility in managing memory is rated on the range and quality of the product's features in optimizing system performance and handling special-case installations. Quality of documentation is judged by the clarity, comprehensiveness and helpfulness of the documentation provided with the product, based on the organization of the manuals as well as the thoroughness of definitions, directions and explanations, and the inclusion of troubleshooting information.

◆ Request 201 on Reader Service Card ◆

That makes *abs* a prime candidate for special treatment as a built-in function generating inline code.

Listing 2 shows the file *div.c*. It provides a portable implementation of the *div* function. You can eliminate the test if you know that negative quotients truncate toward zero. Most computer architectures have a divide instruction that develops both quotient and remainder at the same time. Those that develop proper negative quotients are also candidates for built-in functions. An implementation is at liberty to reorder the members of the structure type *div_t* to match what the hardware generates.

Listing 3 shows the file *labs.c* and Listing 4 shows the file *ldiv.c*. Both define functions that are simply long versions of *abs* and *div*.

The C Standard on the Algorithmic Functions

7.10.2 Pseudo-random sequence generation functions

7.10.2.1 The *rand* function

Synopsis

```
#include <stdlib.h>
int rand(void);
```

Description

The *rand* function computes a sequence of pseudo-random integers in the range 0 to *RAND_MAX*.

The implementation shall behave as if no library function calls the *rand* function.

Returns

The *rand* function returns a pseudo-random integer.

Environmental Limit

The value of the *RAND_MAX* macro shall be at least 32767.

7.10.2.2 The *srand* function

Synopsis

```
#include <stdlib.h>
void srand(unsigned int seed);
```

Description

The *srand* function uses the argument as a seed for a new sequence of pseudo-random numbers to be returned by subsequent calls to *rand*. If *srand* is then called with the same seed value, the sequence of pseudo-random numbers shall be repeated. If *rand* is called before any calls to *srand* have been made, the same sequence shall be generated as when *srand* is first called with a seed value of 1.

The implementation shall behave as if no library function calls the *srand* function.

Returns

The *srand* function returns no value.

Example

The following functions define a portable implementation of *rand* and *srand*.

```
static unsigned long int next = 1;
int rand(void) /* RAND_MAX assumed to be 32767 */
{
    next = next * 1103515245 + 12345;
    return (unsigned int)
        next/65536) % 32768;
}

void srand(unsigned int seed)
{
    next = seed;
}
```

7.10.5 Searching and sorting utilities

7.10.5.1 The *bsearch* function

Synopsis

```
#include <stdlib.h>
void *bsearch(const void *key,
    const void *base, size_t nmemb,
    size_t size, int (*compar
    (const void *, const void *));
```

Description

The *bsearch* function searches an array of *nmemb* objects, the initial element of which is pointed to by *base*, for an element that matches the object pointed to by *key*. The size of each element of the array is specified by *size*.

The comparison function pointed to by *compar* is called with two arguments that point to the *key* object and to an array element, in that order. The function shall return an integer less than, equal to, or greater than zero if the *key* object is considered, respectively, to be



Productivity means Money.
As a software developer, you know that greater productivity means more money in your pocket. Pop-up reference is a positive step toward increased productivity... But only if it's powered by the right Engine. An Engine that runs in 1k of RAM. One that lets you Cut and Paste text directly from your database into your application. This Engine must have Global Searching and Auto-lookup. It must be able to support high resolution VGA monitors. It must also be able to auto-adjust its display window as you move from your 50 line editor into a 24 line Debugger. This Engine must be 100% compatible with the Norton Guides files that 3rd party developers are supplying with their packages. And lastly, you need the tool's to modify text in your existing databases as well as be able to create your own databases.

Introducing... Expert Help.
Expert Help does all of the above and more. The Expert Help System includes the EH Engine and Database Compiler, Linker, DeCompiler, Free incremental upgrades and unlimited Technical Support. With hundreds of databases available and powerful features, Expert Help is the "pop-up" reference system of choice.

Upgrade offer for Norton Guides Users.
For a limited time, Norton Guides users can upgrade to The Expert Help System at a special price. Call for details.

To Order, Call
1-800-325-6820

Australia, Call:
RCM Software, Tel: 61-3-809-2041 Fax: 61-3-889-0263
Europe, Call:
Four Season Software Ltd., Tel: 0279 758022 Fax: 0279 758025
TOBAX Software GmbH, Tel: 0221/738028 Fax: 0221/722806

REPLACES THE NORTON GUIDES.

Features	EH	NG
100% Norton Guides Compatible	✓	✓
Requires 1k of RAM	✓	✓
Cut and Paste Text from Databases	✓	✓
Cut and Print Text from Databases	✓	✓
Searches Entire Databases	✓	✓
Searches Current Menu Item for Information	✓	✓
Auto-Lookup of Information	✓	✓
Auto-Lookup with Global Searching	✓	✓
Supports up to 60 line VGA mode	✓	✓
Mouse Support	✓	✓
Sticky Window (Leaves Image on Screen)	✓	✓
Search Long Entries	✓	✓
Invoke Search at any Level	✓	✓
Configure Colors	✓	✓
Command Line Unload from Memory	✓	✓
Pass-thru Mode	✓	✓
Specify Database at Load Time (Command Line)	✓	✓
20 SeeAlso's, 72 characters long in each Long Entry	✓	✓
SeeAlso's are in a pull down menu	✓	✓
Support for 64k Short and Long Entries	✓	✓
Display Dynamically Adjusts to Screen Size	✓	✓
Quick Keys for Fast Access to Information	✓	✓
Accesses an unlimited number of Databases	✓	✓
Database Compiler/Linker that Supports 64k Entries	✓	✓
Database DeCompiler	✓	✓
Technical Support and Product Upgrades	✓	✓

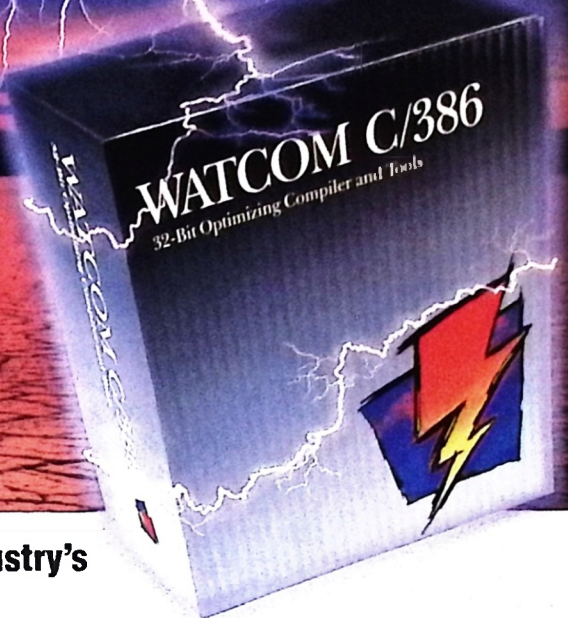
Expert Help
Personal Use Version \$129
LAN Version \$379
Distribution Kit 1 \$329
Distribution Kit 2 \$299

1989 DATA BASED ADVISOR
Readers' Choice Award

SofSolutions
440 Quentin Dr., San Antonio, TX 78201
(512) 735-0746 Fax/BBS (512) 734-8584

Delivering the Power: WATCOM C9.0/386

- ▶ **The Widest Range of 32-bit Intel x86 Platforms**
32-bit DOS, 32-bit Windows, OS/2 2.0, AutoCAD ADS
- ▶ **The Industry's Leading Code Optimizer**
Advanced global optimizer with new 486 optimizations
- ▶ **The Most Comprehensive Toolset**
Debugger, profiler, protected-mode compiler and linker, 32-bit DOS extender with royalty-free run-time, licensed components from Microsoft SDK, and more
- ▶ **The Best Value in 32-Bit Tools: \$895***



Unleash 32-bit Power!

WATCOM C9.0/386 lets you exploit the two key 32-bit performance benefits. The 32-bit flat memory model simplifies memory management and lets applications address beyond the 640K limit. Powerful 32-bit instruction processing delivers a significant speed advantage: typically at least a 2x speedup.

You Get:

- ▶ 100% ANSI and SAA compatible: C9.0/386 passes all Plum Hall Validation Suite tests
- ▶ Extensive Microsoft compatibility simplifies porting of 16-bit code
- ▶ Royalty-free run-time for 32-bit DOS, Windows and OS/2 apps
- ▶ Comprehensive toolset includes debugger, linker, profiler and more
- ▶ DOS extender support for Rational, Phar Lap and Ergo
- ▶ Run-time compatible with WATCOM FORTRAN 77/386

32-bit DOS support includes the DOS/4GW 32-bit DOS extender by Rational Systems with royalty-free runtime license

- ▶ Virtual Memory support up to 32Mb

32-bit Windows support enables development and debugging of true 32-bit GUI applications and DLLs.

- ▶ Includes licensed Microsoft SDK components

32-bit OS/2 2.0 support includes development for multiple target environments including OS/2 2.0, 32-bit DOS and 32-bit Windows

- ▶ Access to full OS/2 2.0 API including Presentation Manager
- ▶ Integrated with IBM Workframe/2 Environment

AutoCAD ADS and ADI Development: Everything you need to develop and debug ADS and ADI applications for AutoCAD Release 11

Novell's *Network C* for NLM's SDK includes C/386

The Industry's Choice.

Autodesk, Robert Wenig, Manager, AutoCAD for Windows:

"At Autodesk, we're using WATCOM C/386 in the development of strategic new products since it gives us a competitive edge through early access to new technologies. We also highly recommend WATCOM C/386 to third party AutoCAD add-on (ADS and ADI) developers."

Fox Software, David Fulton, President: "FoxPro 2.0 itself is written in WATCOM C, and takes advantage of its many superior features. Optimizing for either speed or compactness is not uncommon, but to accomplish both was quite remarkable."

GO, Robert Carr, Vice President of Software: "After looking at the 32-bit Intel 80x86 tools available in the industry, WATCOM C was the best choice. Key factors in our decision were performance, functionality, reliability and technical support."

IBM, John Soyryng, Director of OS/2 Software Developer Programs: "IBM and WATCOM are working together closely to integrate these compilers with the OS/2 2.0 Programmer's Workbench."

Lotus, David Reed, Chief Scientist and Vice President, Pen-Based Applications: "In new product development we're working with WATCOM C because of superior code optimization, responsive support, and timely delivery of technologies important to us like p-code and support for GO Corp's. PenPoint."

Novell, Nancy Woodward, V.P. and G.M., Development Products: "We searched the industry for the best 386 C compiler technology to incorporate with our developer toolkits. Our choice was WATCOM."

WATCOM

1-800-265-4555

The Leader in 32-bit Development Tools

415 Philip Street, Waterloo, Ontario, Canada. Telephone: (519) 866-3700, Fax: (519) 747-4971. *Price does not include freight and taxes where applicable. Authorized dealers may sell for less. WATCOM C and Lightning Device are trademarks of WATCOM Systems Inc. DOS/4GW and DOS/16M are trademarks of Rational Systems Inc. Other trademarks are the properties of their respective owners. Copyright 1992 WATCOM Products Inc.



◆ Request 126 on Reader Service Card ◆

less than, to match, or to be greater than the array element. The array shall consist of: all the elements that compare less than, all the elements that compare equal to, and all the elements that compare greater than the *key* object, in that order.¹²⁹ [FN129. In practice, the entire array is sorted according to the comparison function.]

Returns

The *bsearch* function returns a pointer to a matching element of the array, or a null pointer if no match is found. If two elements compare as equal, which element is matched is unspecified.

7.10.5.2 The *qsort* function

Synopsis

```
#include <stdlib.h>
void qsort(void *base, size_t nmem, size_t size,
           int (*compar)(const void *, const void *));
```

Description

The *qsort* function sorts an array of *nmem* objects, the initial element of which is pointed to by *base*. The size of each object is specified by *size*.

The contents of the array are sorted into ascending order according to a comparison function pointed to by *compar*, which is called with two arguments that point to the objects being compared. The function shall return an integer less than, equal to, or greater than zero if the first argument is considered to be respectively less than, equal to, or greater than the second.

If two elements compare as equal, their order in the sorted array is unspecified.

Returns

The *qsort* function returns no value.

Using the Algorithmic

Functions

Here is a brief summary of the four algorithmic functions:

bsearch — Use this function to search any array whose elements are ordered by pairwise comparisons. You define the ordering with a comparison function that you provide. For example, you can build a keyword lookup function from the basic form as shown in Listing 5.

A few caveats:

- If a key compares equal to two or more elements, *bsearch* can return a pointer to any of these elements.
- Beware of changes in how elements sort when the execution character set changes — call *qsort*, described below, with a compatible comparison function to ensure that an array is properly ordered.
- Be careful using the functions *strcmp* or *strcoll*, declared in *<string.h>*, directly. Both require that strings be stored in the array to be searched. You cannot use them to search an array of pointers to strings. To use *strcmp*, for example, you must write a function pointer argument that looks like `(int (*)(const void *, const void *))&strcmp`.

qsort — Use this function to sort any array whose elements are ordered by pairwise comparisons. You define the ordering with a comparison function that you provide. The comparison function



Simple. Elegant. Reliable.

Features

- Priority based scheduler with time slicing and preemption options
- Unlimited number of tasks, queues, resources, and events
- Fixed and Variable memory management
- Time-out option on all service calls
- ROMable C source code included
- No royalties
- MS-DOS compatible file system

Processors Supported

80x86s, 80386/486 protected mode, 68xxxs, 683xxs, Am29000 RISC family, Intel's i960, SPARClite, MIPS, TMS320C3x and the TMS320C2x.

NUCLEUS
RTX

Real-time Multi-tasking Executive

Simple solutions are generally better solutions. Complicated solutions often reflect poor design. Nucleus embodies simplicity. Its elegant style and design make it easy to use and understand. Nucleus is the best, most reliable choice for your embedded application.

Let us prove it.
Call now for more information.

(800) 468-NUKE



ACCELERATED TECHNOLOGY INC.
P.O. Box 850245 Mobile, AL 36685

(205) 450-0707

has a specification similar to that for the function *bsearch*, described above. Note, however, that the *bsearch* comparison function compares a key to an array element. The *sort* comparison function compares two array elements.

A few caveats:

- Don't assume that the function uses the "Quicksort" algorithm, despite the name. It may not. If two or more elements compare equal, *qsort* can leave these elements in any relative order. Hence, *qsort* is not a *stable* sort.
- Beware of changes in how elements sort when the execution character set changes.
- Be careful using the functions *strcmp* or *strcoll* declared in *<string.h>*, directly. Both require that strings be stored in the array to be sorted. You cannot use them to sort an array of pointers to strings. To use *strcmp*, for example, you must write a function pointer argument that looks like *(int (*)(const void *, const void *))&strcmp*.

rand — Call *rand* to obtain the next value in a pseudo-random sequence. You get exactly the same sequence following each call to *srand* with a given argument value. That is often desirable behavior, particularly when you are debugging a program. If you want less predictable behavior, call *clock* or *time*, declared in *<time.h>* to obtain an argument for *srand*. The behavior of *rand* can vary among implementations. If you want exactly the same pseudo-random sequence at all times, copy the version presented here.

Use *RAND_MAX* to scale values returned from *rand*. For example, if you want random numbers of type *float* distributed over the interval [0.0, 1.0], write the expression *(float)rand()/RAND_MAX*. The value of *RAND_MAX* is at least 32,767.

srand — See *rand* above. The program effectively calls *srand(1)* at program startup.

Implementing the Algorithmic Functions

Listing 6 shows the file *qsort.c*. It defines the related function *qsort* that sorts an array beginning at *base*. I introduced the type *_Cmpfun* just to simplify the declaration of arguments for the functions *bsearch* and *qsort*. Don't use this declaration in code that you write if you want it to be portable to other implementations.

This logic is much less simple and more debatable. It is based on the Quicksort algorithm first developed by C.A.R. Hoare. That requires you to pick a partition element, then partially sort the array about this partition. You can then sort each of the two partitions by recursive application of the same technique. The algorithm can sort quite rapidly. It can also sort very slowly.

How best to choose the pivot element is the debatable issue. Pick the first element and an array already in sort eats a lot of time. Pick the last element and an array in reverse sort eats a lot of time. Work too hard at picking an element and all arrays eat a lot of time. I chose simply to pick the last element. That favors arrays that need little rearranging. You may have reason to choose another approach.

qsort calls itself to sort the smaller of the two partitions. It loops internally to sort the larger of the two. That minimizes demands on dynamic storage. At worst, each recursive call must sort an array half as big as the earlier call. To sort *N*

Listing 5

```
#include <stdlib.h>
#include <string.h>

typedef enum {FLOAT, INTEGER} Code;
typedef struct {
    char *s;
    Code code;
} Entry;
Entry symtab[] = {
    {"float", FLOAT},
    {"integer", INTEGER}}

static int cmp(const void *ck, const void *ce)
{
    /* compare key to table element */
    return (strcmp((const char *)ck, ((Entry *)ce)->s));
}

Entry *lookup(char *key)
{
    /* lookup key in table */
    return (bsearch(key, symtab,
        sizeof symtab / sizeof symtab[0],
        sizeof symtab[0], &cmp));
}

/* End of File */
```

elements requires recursion no deeper than $\log_2(N)$ calls. (You can sort 1,000,000 elements with at most 20 recursive calls.)

Listing 7 shows the file *bsearch.c*. The function *bsearch* performs a binary search on the sorted array beginning at *base*. The logic is simple but easy to get wrong.

WHY RE-INVENT THE WHEEL?

**CONTROL
BLOK**

We just finished writing half
of your automation software
for you

- ☐ **PID Control Loops**
- ☐ **Multi-drop RS485**
- ☐ **Async RS232**
- ☐ **Remote OPTO I/O**
- ☐ **BITBUS™ Network**
- ☐ **DIVVY™ Multi-tasking**

The common functions you need are available in pre-coded, pre-tested ControlBLOK™ software modules. Just pick the modules you need, then add your own code. You'll save weeks of development and debug time. Call or FAX today for ControlBLOK™ details.

Drumlin

1011 Grand Central Ave. • Glendale, CA 91201
FAX (818) 244-4246 • Phone (818) 244-4600

ControlBLOK, DIVVY™, Drumlin Other products™ their manufacturers

Listing 6 (qsort.c)

```

/* qsort function */
#include <stdlib.h>
#include <string.h>

/* macros */
#define MAX_BUF256 /* chunk to copy on swap */

void (qsort)(void *base, size_t n, size_t size, _Cmpfun *cmp)
{ /* sort (char base[size])[n] using quicksort */
    while (1 < n)
    { /* worth sorting */
        size_t i = 0;
        size_t j = n - 1;
        char *qi = (char *)base;
        char *qj = qi + size * j;
        char *qp = qj;

        while (i < j)
        { /* partition about pivot */
            while (i < j && (*cmp)(qi, qp) <= 0)
                ++i, qi += size;
            while (i < j && (*cmp)(qp, qj) <= 0)
                --j, qj -= size;
            if (i < j)
            { /* swap elements i and j */
                char buf[MAX_BUF];
                char *q1 = qi;
                char *q2 = qj;
                size_t m, ms;

                for (ms = size; 0 < ms; ms -= m, q1 += m, q2 -= m)
                { /* swap as many as possible */
                    m = ms < sizeof (buf) ? ms : sizeof (buf);
                    memcpy(buf, q1, m);
                    memcpy(q1, q2, m);
                    memcpy(q2, buf, m);
                }
            }
            j = n - i - 1, qi += size;
            if (j < i)
            { /* recurse on smaller partition */
                if (1 < j)
                    qsort(qi, j, size, cmp);
                n = i;
            }
            else
            { /* lower partition is smaller */
                if (1 < i)
                    qsort(base, i, size, cmp);
                base = qi;
                n = j;
            }
        }
    }
}

/* End of File */

```

Liana™

Interpretive C-like Object-Oriented Programming Language and Class Library for Windows 3

- Great for casual programming, prototyping
- Syntax of C++ simplified and extended
- Faster and less error-prone coding
- Automatic memory management
- Simplified string manipulation
- Flexibility of an interpreter
- Better run-time error detection
- High level class library
- DDE and DLL support
- Low cost **Personal Developer: \$129**
- Royalty-free **Professional Developer: \$495**

Base Technology 1543 Pine St., Boulder, CO 80302
800-786-9505 303-440-4558 Compuserve [70642,2662]

◆ Request 204 on Reader Service Card ◆

debug RS-232

with a low cost DataScope™

- Alterable character sets
- Macro display recording
- Up to four user-alterable, multitasking display windows
- Microsecond timestamps
- Laptop color support
- Free demo diskette
- Hypertext help



Only \$299!
Includes cable,
manual and
support.

"DataScope has been indispensable in our protocol work!" - FORTH, Inc.

Paladin Software, Inc.

800-397-1368

◆ Request 106 on Reader Service Card ◆

NOTICE

WE'VE BEEN ASKED TO MAKE IT VERY CLEAR THAT
THERE'S A WHOLE OF A DIFFERENCE BETWEEN QNX[®] AND UNIX[®].

WE'RE GLAD TO OBLIGE.

EVEN THOUGH THE QNX REALTIME OPERATING SYSTEM IS *NOT* BASED ON UNIX SOURCE CODE, WE CAN UNDERSTAND HOW SOME DEVELOPERS MIGHT CONFUSE THE TWO. AFTER ALL, QNX FOLLOWS THE LATEST IEEE POSIX 1003.1 AND 1003.2 OPEN SYSTEMS STANDARDS, SO YOU GET THE SAME API AND UTILITY SET FOUND IN MANY UNIX SYSTEMS.

BUT LOOK BENEATH THE SURFACE AND YOU'LL SEE TWO FUNDAMENTALLY DISTINCT ARCHITECTURES. A MONOLITHIC OS ON THE ONE HAND, A MICROKERNEL OS ON THE OTHER. THEY'RE DIFFERENT SPECIES ALTOGETHER, AS DIFFERENT AS A WHALE AND A SCHOOL OF DOLPHINS.

ARCHITECTURE YOU CAN BUILD ON
QNX'S MODULAR, MICROKERNEL ARCHITECTURE GIVES YOU REMARKABLE FLEXIBILITY. OS SERVICES ARE PROVIDED BY A TEAM OF COOPERATING PROCESSES RATHER THAN BY A MASSIVE MONOLITHIC KERNEL. THE RESULT? YOU CAN STRIP QNX DOWN TO A ROM-ABLE EMBEDDED SYSTEM. OR BUILD IT UP TO A VAST NETWORK AND HARNESS THE POWER OF HUNDREDS OF CPUS.

OUR PRICING IS MODULAR TOO, SO YOU'RE NOT FORCED TO PAY FOR WHAT YOU DON'T NEED.

REALTIME PERFORMANCE

YOU CAN COUNT ON

NEED THE SPEED OF A FAST, REALTIME EXECUTIVE?
QNX CLOCKS IN AT 16 μ SEC PER CONTEXT SWITCH ON A 33 MHZ 80486.

BUT QNX DOESN'T STOP THERE. WITH ITS PRIORITY-DRIVEN, PREEMPTIVE SCHEDULING, YOU CAN BUILD *REAL* REALTIME SOLUTIONS.

DISTRIBUTED PROCESSING

YOU CAN BET ON

QNX CAN TRANSFORM A BUNCH OF ISOLATED MACHINES INTO A SEAMLESS SUPERCOMPUTER,



REALTIME OPERATING SYSTEM

ORCHESTRATING HUNDREDS OF CPUS WITH ITS NETWORK-WIDE IPC.

YOU'RE IN COMPLETE CONTROL OF ALL RESOURCES AT ANY POINT, FROM THE PLANT FLOOR RIGHT UP TO YOUR DESKTOP.

SUPPORT YOU CAN DEPEND ON

DURING THE LAST TEN YEARS, WE'VE EARNED A REPUTATION FOR OUTSTANDING TECHNICAL SUPPORT. WE OFFER EVERYTHING FROM 24-HOUR ONLINE CONFERENCING TO ONSITE CONSULTING, SO YOU CAN EASILY REACH THE PEOPLE WHO MAKE UP THE QNX DEVELOPMENT TEAM. THEIR EXPERTISE CAN HELP YOU KEEP YOUR DEVELOPMENT PROJECTS RIGHT ON COURSE.

TO FIND OUT HOW YOUR APPLICATIONS CAN THRIVE IN THE QNX ENVIRONMENT, CALL 1-800-363-9001, (EXT. 102).

QUANTUM SOFTWARE SYSTEMS LTD.
175 TERENCE MATTHEWS CRESCENT
KANATA, ONTARIO, CANADA
K2M 1W8
TEL: 613-591-0931
FAX: 613-591-3579

ARCHITECTURE MAKES THE DIFFERENCE

QNX IS A REGISTERED TRADEMARK OF QUANTUM SOFTWARE SYSTEMS LTD. UNIX IS A REGISTERED TRADEMARK OF UNIX SYSTEM LABORATORIES, INC. © QUANTUM SOFTWARE SYSTEMS LTD. 1991

◆ Request 478 on Reader Service Card ◆

Listing 7 (bsearch.c)

```
/* bsearch function */
#include <stdlib.h>

void *(bsearch)(const void *key, const void *base,
    size_t nelem, size_t size, _Cmpfun *cmp)
{ /* search sorted table by binary chop */
    const char *p;
    size_t n;

    for (p = (const char *)base, n = nelem; 0 < n; )
    { /* check midpoint of whatever is left */
        const size_t pivot = n > 1;
        const char *const q = p + size * pivot;
        const int val = (*cmp)(key, q);

        if (val < 0)
            n = pivot; /* search below pivot */
        else if (val == 0)
            return ((void *)q); /* found */
        else
        { /* search above pivot */
            p = q + size;
            n -= pivot + 1;
        }
    }

    return (NULL); /* no match */
}

/* End of File */
```

Listing 8 (rand.c)

```
/* rand function */
#include <stdlib.h>

/* the seed */
unsigned long _Randseed = 1;

int (rand)(void)
{ /* compute pseudo-random value */
    _Randseed = _Randseed * 1103515245 + 12345;
    return ((unsigned int)(_Randseed >> 16) & RAND_MAX);
}

/* End of File */
```

Listing 9 (srand.c)

```
/* srand function */ #include <stdlib.h>

void (srand)(unsigned int seed)
{ /* alter the seed */
    _Randseed = seed;
}

/* End of File */
```

Listing 8 shows the file *rand.c*. The function *rand* generates a pseudo-random sequence using the algorithm suggested in the C Standard. That has reasonable properties, plus the advantage of being widely used. One virtue of a random number generator is randomness. Another virtue, ironically, is reproducibility. You often need to check that a calculation based on pseudo-random numbers does what you expect. The arithmetic is performed using *unsigned long* integers to avoid overflows.

Listing 9 shows the file *srand.c*. The function *srand* simply sets *_Randseed*, the seed for the pseudo-random sequence generated by *rand*. I provide a masking macro for *srand*. Hence, the header *<stdlib.h>* declares *_Randseed*, defined in *rand.c*.

References

Donald Knuth, *The Art of Computer Programming*, Vols. 1-3 (Reading, Mass.: Addison-Wesley, 1967 and later). Here is a rich source of algorithms, complete with analysis and tutorial introductions. Volume 1 is *Fundamental Algorithms*, volume 2 is *Seminumerical Algorithms*, and volume 3 is *Sorting and Searching*. Some are in second edition.

You will find oodles of information on:

- maintaining a heap
- computing random numbers
- searching ordered sequences
- sorting
- converting between different numeric bases

Before you tinker with the code presented here, see what Knuth has to say. □

This article is excerpted in part from P.J. Plauger, The Standard C Library, (Englewood Cliffs, N.J.: Prentice-Hall, 1992).

Save time, cut code size, and manage objects efficiently with

Style™ "The Elegant Solution"
for C++

Style is a general-purpose class library that provides high-level object management for C++. There is no other tool like it.

Your C++ assistant

With Style, the implementation of associations and links among objects becomes transparent. By removing many of the trivial details of C++ programming, Style helps you reduce your development effort by up to 75%.

Redefining "power"

High-level object management is only one feature of Style. Powerful traversal functions selectively navigate through the objects in your application and perform operations on those objects. You specify what types of objects to target; the traversal functions perform the leg-work.

Rapid application development, more maintainable applications, less application code — shouldn't you code with Style?

Supports most C++ compilers. Available for MS-DOS, Windows, and UNIX.



List price \$250
30-day money back guarantee
Source code available

P.O. Box 1586
Ballwin, MO 63022
Tel: (314) 391-7772
Fax: (314) 391-0727

♦ Request 170 on Reader Service Card ♦

Multiple Copy Math Functions

Dr. Timothy Prince

Pipelined architectures including vector and superscalar obtain their speed by depending in part on working on independent calculations in pipeline fashion. Most computationally intensive applications present enough opportunities for parallel or pipeline operation to make worthwhile increases in speed. The normal use of scalar math functions, which produce a single result, poses an obstacle to superscalar performance. These functions can be organized to increase the internal opportunities for parallelism as compared with optimum scalar processor code. Performance remains far short of the potential, unless more parallelism is exploited by working on more than one math function result at a time. A further reason for obtaining multiple results is that the overhead for calling functions which take less than 10 microseconds becomes excessive.

Soon after the introduction of vector computers, vector math functions were introduced to provide vector performance in calculations involving such functions. With pipelined or superscalar processors, vector functions may be effective, but functions which calculate a small number of copies per call are more versatile. Since a typical RISC architecture employs a four-stage pipeline, functions which calculate two or four copies should be enough to maximize performance.

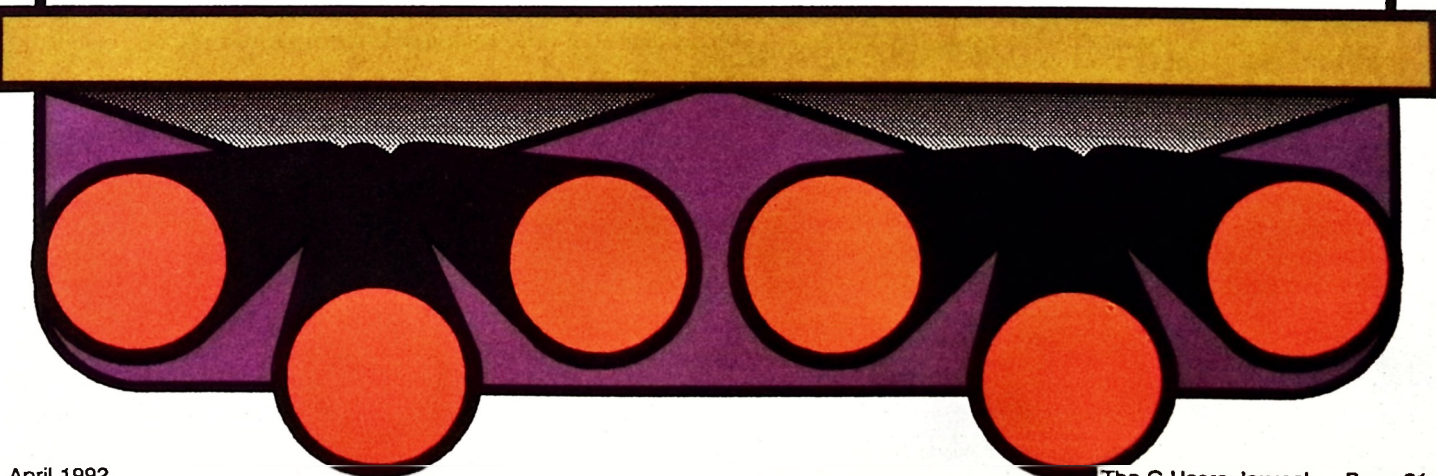
Vector chunk math functions may be used whether or not your compiler makes specific provision for them. I will show actual examples of the C code of such functions. The functions *sin_4* (four *sins*), *cosf_sinf_2* (two pairs of *float sin* and *cos*), *powf_2* (two *float pows*), and *tan_2* (two *tans*) are chosen for their proven usefulness and because they illustrate the points which I want to make.

Vector vs Superscalar Function Calls

On a vector architecture, vector math functions naturally are performed on argument vectors, and normal vector performance is not approached except on long vectors. These architectures perform well when 50 or more functions are to be calculated at a time. Suppose we wanted to integrate a function involving *sin* and *cos* by Simpson's rule, producing a loop such as

```
for(i=2 ; i<n ; i += 2){
    yint += q[i-2]*sin(t[i-2])
        +4*q[i-1]*sin(t[i-1])+q[i]*sin(t[i]);
    xint += q[i-2]*cos(t[i-2])
        +4*q[i-1]*cos(t[i-1])+q[i]*cos(t[i]);
}
```

Timothy Prince has a B.A. in physics from Harvard and a Ph.D. in mechanical engineering from the University of Cincinnati. He has 25 years of experience in aerodynamic design and computation. He can be contacted at 452 Palmitas St., Solana Beach, CA 92075.



Listing 1 (s)

```

/*
** bc program to calculate Chebyshev economized polynomial
** for evaluation of sin(x) */
/* use bc -l to get c() and s() functions */
define t(x) {
    /* sin(x)/x */
    if(x==0) return(1.); /* derivative of s function */
    return (s(x) / x); /* put function to be fit here */
}
define b(x) {
    if (x < 0) return (-x);
    return (x);
}
define m(x, y) {
    if (x > y) return (x);
    return (y);
}
n = 22; /* number of Chebyshev terms */
scale = 40;
p = a(1.) * 4; /* pi */
b = p * .5; /* upper end of curve fit interval */
a = -b; /* lower end of interval */
/* chebft adapted from Press Flannery et al */
/* "Numerical Recipes" FORTRAN version */
for (k = 1; k <= n; ++k) {
    c[k] = 0;
    f[k] = t(c((k - .5) * p / n) * (b - a) * .5 + (b + a) * .5);
}
/* because of symmetry, even c[] are 0 */
for (j = 1; j <= n; j += 2) {
    s = 0;
    q = (j - 1) * p / n;
    for (k = 1; k <= n; ++k) s += c(q * (k - .5)) * f[k];
    (c[j] = 2 / n * s);
}

/* skip even terms, which are 0 */
for (n = 5; n <= 19; n += 2) {
    /* chebpc */
    for (j = 1; j <= n; ++j) d[j] = e[j] = 0;
    d[1] = c[n];
    for (j = n - 1; j >= 2; --j) {
        for (k = n - j + 1; k >= 2; --k) {
            s = d[k];
            d[k] = d[k - 1] * 2 - e[k];
            e[k] = s;
        }
        s = d[1];
        d[1] = c[j] - e[1];
        e[1] = s;
    }
    for (j = n; j >= 2; --j) d[j] = d[j - 1] - e[j];
    d[1] = c[1] * .5 - e[1];
    /* pcshft */
    g = 2 / (b - a);
    for (j = 2; j <= n; ++j) {
        d[j] = g;
        g = 2 / (b - a);
    }
    for (j = 1; j <= n; ++j) {
        h = d[n];
        for (k = n - 1; k >= j; --k) {
            h = d[k] - (a + b) * .5 * h;
            d[k] = h;
        }
    }
    /* Chebyshev Sin fit |x| <= PI/2 coefficients */
    /* Maximum Rel Error: */
    m(b(c[n + 2]), b(c[2])) / t(b);
    for (i = 1; i <= n; i += 2) d[i];
}
/* End of file */

```

which involves n evaluations of \sin and \cos . A vector compiler would start out by setting up the six vectors made up of the three \sin and \cos evaluations from each copy of the loop body. Almost a third of these evaluations would be duplicates, since the vector of $\sin(t[i-2])$ is the same as the vector of

$\sin(t[i])$ except that $\sin(t[0])$ and $\sin(t[n-1])$ are not repeated. Each of these vectors has length $(n+1)/2$, so n would have to be around 100 before good vector performance could be achieved.

In order to approach the performance potential of a vector architecture, we would have to rewrite the code to store the $q[]*\sin()$ and $q[]*\cos()$ intermediate results in vectors in a preliminary loop, and then add the appropriate values in another loop. Even after 20 years of vector compiler development, this is more analysis than any compiler can do without human assistance. Most reasonable attempts to improve the performance of this loop for a scalar architecture will prevent vectorization, and changes to improve vector performance will reduce scalar or superscalar performance. Although vector compilers now deal well with sum reductions such as this loop, this is done in effect by splitting the vectors again into six to 10 shorter vectors, making a vector architecture less than fully effective for this type of application.

Unrolling compilers can eliminate most of the duplicate operations by combining the common subexpressions over several iterations of the loop. Each additional loop iteration will require an additional pair of \sin s and \cos s. Compilers have been available (e.g. Multi-flow) which would detect this situation automatically and build in a call such as $\cosf_sinf_2(t[0], t[1])$ which returns



MultiTask!™ Execs and GOFAST™ Math Speed Time-to-Market

Zap your application with peak performance using tested code and expert support. Control real-time scheduling with MultiTask! source code executives, or ROM-able, re-entrant GOFAST floating point:

- Replace 80x87 MATH COPROCESSORS;
- Drop-in IEEE SOURCE LIBRARIES;
- Link-and-Go with C compilers: Intel®, Microsoft®, Borland®, WATCOM®, Zortech, Metaware®, and more...

Solutions for 80386/486 PROTECTED-MODE, 80x86 & V-Series, Z80/180/64180, 8085, 68xxx, 68HC16, 68HC11, 6801, 6809, 8051, 80196, i960, R3000, SPARC® and more.

Call for free information diskettes today.
PHONE 503-641-8446; FAX 503-644-2413;
USA TOLL FREE 800-356-7097.



14215 NW Science Park Drive
Portland, OR 97229

U S SOFTWARE®

© 1992 US Software Corporation. GOFAST and MultiTask! are trademarks of US Software Corporation. All other trademarks belong to their respective owners.

◆ Request 476 on Reader Service Card ◆

cosf and *sinf* of both arguments, a total of four results from one function call. Such a function is a good match to the architecture of a superscalar processor. Even if your compiler does not perform the transformation automatically, manual rewriting is not unduly burdensome and need not detract from performance on scalar processors. A change as simple as

```
ty = q[0]*sinf(t[0]);
tx = q[0]*cosf(t[0]);
for( i=2; i<n; i+=2){
    temp = cosf_sinf_2(t[i-1],t[i]);
    yint += ty+q[i]*temp.sin2+4*q[i-1]*temp.sin1;
    xint += tx+q[i]*temp.cos2+4*q[i-1]*temp.cos1;
    tx = q[i]*temp.cos2;
    ty = q[i]*temp.sin2;
}
```

should produce most of the advertised performance of any non-vector machine.

Vector and Vector Chunk Function Coding Style

Multiple copy or vector chunk math functions, like vector code, need to be written without conditionals which actually cause transfer of control. In a scalar trig function, it would usually be worth while to test the argument to find out whether it needs to be translated into the primary range. In a vector chunk function, the full range reduction should be performed whether it is needed or not, so that all of the code for the function can be compiled as one basic block.

Transfer of control (branching) gives the compiler a choice of undesirable consequences. Either the pipelines must be allowed to empty, reducing the performance to scalar levels until they are refilled, or *trace scheduling* must be used to fill the pipeline with future operations along the preferred path of execution. Each branch can cause generation of another trace, and the length of compiled code may grow exponentially with the number of branches. The speed gained by keeping the pipelines full may be canceled by increased paging.

A great deal of progress has been made in architecture and compilers in recent years, so that many simple conditional selections can be performed without a transfer of control, if this is necessary to keep a processor producing results at rated speed. This requires calculation of both alternative results followed by instructions which select the correct one. There is a good correspondence with the syntax of *?:* in C, although the compiler should not be totally dependent on the programmer choosing to use *?:*. Existing compilers do not vectorize *if..else*. All of the operations in the *sin*, *cos*, *tan*, *exp*, and *log* functions can and should be written

in vectorizable form, even when the overall scheme is to favor superscalar execution.

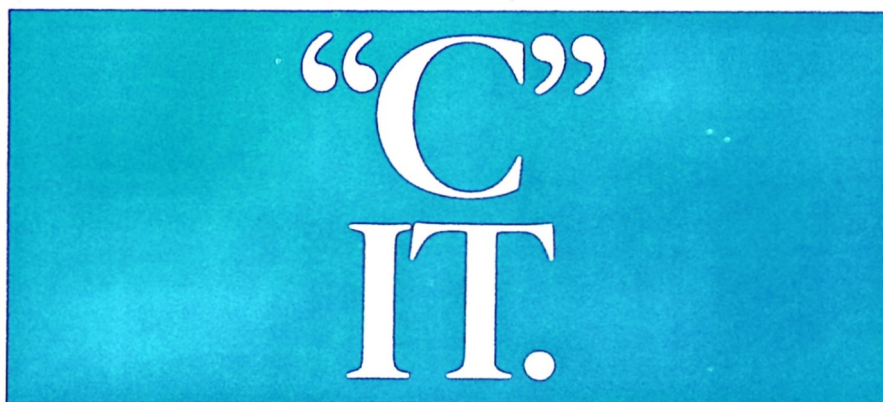
Vector chunk functions do not fit well with the *<errno.h>* system for error reporting. The best that can be done is to report that *ERANGE* or *EDOM* exceptions have been raised for one of the arguments or results. Vector functions give an even hazier indication of trouble.

Some Nuts and Bolts of Machine Dependence

In some of the examples, the sign of a *float* or *double* is tested by assuming that it is in the same position as the sign of an *int* which shares the same storage. This is done either because it is faster or because it reduces register thrashing on certain machines. Generally, in these functions, there is an imbalance of *double* over integer arithmetic, and integer operations can be treated as a free resource whenever *float* operations are being performed at the same time.

This code will work as is on most modern architectures which use the same byte order for *double* and *int*. On VAX-compatible architectures, the sign of a *double* falls in place with the sign of a *short* at the same address, apparently as a result of the PDP-11 ancestry.

A few architectures also suffer from excess of divide and multiply operations over add and subtract, so, in the examples, addition is used instead of multiply by 2. In these examples, it occurs when there are plenty of pipeline slots open, but in other cases, one would not want to prevent an optimizing compiler from converting multiply by 2 to a *ldexp* operation.



© 1992 AT&T

RC-EASY.

Imagine freedom from restrictive, costly proprietary 4GLs. The power to build sophisticated data entry screens and access them from within your "C" programs. And the portability of moving applications between Unix and MS-DOS without changing code.

RC-EASY provides exactly that—the freedom, power and portability to reinforce "C" with popular 4GL features. A 4GL-like environment enables fast application prototyping so that data entry screens and menus can be created, usually without compiling. RC-EASY also features a cohesive set of high-level "C" functions which enables even the most novice programmer to design powerful data entry applications adaptable to any database.

If you're looking for proven performance and reliability in your software, call 1 800 462-8146 for our free catalog.

UNIX is a registered trademark of U.S.L. MS-DOS is a registered trademark of Microsoft Corp.

THE INTELLECTUAL PROPERTY DIVISION

Innovation you can depend on.

◆ Request 150 on Reader Service Card ◆



Since a good pipelining compiler will give priority to finishing up the expressions which are placed first in the code, the

[illegible]

```
typedef struct {
    double X1, X2, X3, X4;
} ARG_D 4; /* vector 4 */
#include "float.h"
ARG_D 4 sin_4(xx1, xx2, xx3, xx4)
    double xx1, xx2, xx3, xx4;
/* use where cos of same argument not needed
** 16 digits precision, compare to 15 digits in "dtrig.h"
** T C Prince */
{
    union dblfmt {
        double dbl;
        int idbl;
        struct dfmt { /* IEEE p754 */
            unsigned int sign:1;
            unsigned int ex:11;
        } fmt;
    } xil;
    double xr, x2, x4, x8;
#ifdef STDC
    long double pi = 3.1415926535897932385, pml;
#include <limits.h>
#else
    register double pi = 3.1415926535897932385, pml;
#define LONG_MIN 0x80000000
#endif
    union dblfmt pm, round;
    ARG_D 4 res;
#define BIAS DBL_MAX_EXP
#include <errno.h>
#ifdef errno
    extern int errno;
#endif
#define ODD(i) ((i)&1)
/* use identity  $\sin(x + n \pi) = (-1)^n \sin(x)$ 
** to reduce range to  $-\pi/2 < x < \pi/2$ 
**  $pml = \text{rint}(xil/pi)$  */
#if FLT_ROUNDS != 1
    #error "rounding mode not nearest; adjust code"
#endif
#if FLT_RADIX != 2 && FLT_RADIX != 10
    #error "code not optimum for accuracy in this RADIX"
#endif
#if DBL_DIG > 16
    #error "more terms needed for full accuracy"
#endif
/* shortcut test of sign, not portable to VAX */
    round.dbl = 1 / LDL_EPSILON;
    xil.dbl = xx1;
    round.idbl |= xil.idbl & LONG_MIN;
    pml = xx1 / pi + round.dbl;
/* sign reversal may reduce register usage */
    xr = pi * (pml == round.dbl) - xx1;
/* shortcut test for fabs(pml) > INT_MAX */
    pm.dbl = pml;
    if (pm.fmt.ex > BIAS + 31)
        errno = ERANGE;
/* don't wait to calculate xr**2 until sign is fixed;
** another sign reversal is due if pm.dbl is odd */
    x2 = xr * xr;
/* first sign reversal compensated in coefficient signs;
** conditional sign fixed by testing odd/even
** first two results are obtained by straight Horner
** polynomial evaluation */
```

```

res.X1 = (-.9999999999999999 + x2 * (.1666666666666607
+ x2 * (-.833333333328281e-2 + x2 * (.19841269824875e-3
+ x2 * (-.2755731661057e-5 + x2 * (.25051882036e-7
+ x2 * (-.160481709e-9 + x2 * .7374418e-12))))))
* (ODD((int) pm.db1) ? -xr : xr);

/* sin(xi2) */
round.db1 = 1 / LOBL_EPSILON;
x1l.db1 = xx2;
round.idbl |= x1l.idbl & LONG_MIN;
pml = xx2 / pi + round.db1;
xr = pi * (pml -= round.db1) - xx2;
pm.db1 = pml;
if (pm.fmt.ex > BIAS + 31)
    errno = ERANGE;
x2 = xr * xr;
res.X2 = (-.9999999999999999 + x2 * (.1666666666666607
+ x2 * (-.833333333328281e-2 + x2 * (.19841269824875e-3
+ x2 * (-.2755731661057e-5 + x2 * (.25051882036e-7
+ x2 * (-.160481709e-9 + x2 * .7374418e-12))))))
* (ODD((int) pm.db1) ? -xr : xr);

/* sin(xi3) */
round.db1 = 1 / LOBL_EPSILON;
x1l.db1 = xx3;
round.idbl |= x1l.idbl & LONG_MIN;
pml = xx3 / pi + round.db1;
xr = pi * (pml -= round.db1) - xx3;
pm.db1 = pml;
if (pm.fmt.ex > BIAS + 31)
    errno = ERANGE;
x2 = xr * xr;
x4 = x2 * x2;

/* split into 2 Horner polynomials to increase
** parallelism after 1st result finishes */
res.X3 = (-.9999999999999999 + x2 * (.1666666666666607
+ x2 * (-.833333333328281e-2
+ x2 * .19841269824875e-3))
+ (-.2755731661057e-5 + x2 * (.25051882036e-7
+ x2 * (-.160481709e-9
+ x2 * .7374418e-12))) * x4 * x4) *
(ODD((int) pm.db1) ? -xr : xr);

/* sin(xi4) */
round.db1 = 1 / LOBL_EPSILON;
x1l.db1 = xx4;
round.idbl |= x1l.idbl & LONG_MIN;
pml = xx4 / pi + round.db1;
xr = pi * (pml -= round.db1) - x1l.db1;

/* errno is set to ERANGE if any of the arguments are too
** large for reasonable range reduction */
pm.db1 = pml;
if (pm.fmt.ex > BIAS + 31)
    errno = ERANGE;
x2 = xr * xr;
x4 = x2 * x2;
x8 = x4 * x4;

/* multiply by 1 is K&R way to enforce parentheses */
res.X4 = ((-.9999999999999999 + x2 * (.1666666666666607
+ x2 * (-.833333333328281e-2
+ x2 * .19841269824875e-3)) * 1
+ (-.2755731661057e-5 + x2 * (.25051882036e-7
+ x2 * (-.160481709e-9 + x2 * .7374418e-12))) * x8)
* (ODD((int) pm.db1) ? -xr : xr);

return res;
}

/* End of File */

```


The later copies are written for more available parallelism at the expense of register usage, so that, when the calculation of the earlier results has finished, the pipelines can still be kept full nearly to the end of the function. This can lead to somewhat greater round off errors in the later copies, in the *double* functions. In the *float* functions, use of *double* arithmetic eliminates the effect of order of operations on accuracy.

Systems which are unable to perform simple conditional selections without branching may require sign changes to be performed by *xoring* the sign bit. To avoid branching, *errno* may be left alone or set by

```
errno=ERANGE&(-(relational expression))
```

which sets it to zero or to *ERANGE*. This is contrary to the normal requirement that *errno* never be set to zero, but may be a satisfactory compromise.

Calculation of Coefficients

Listing 1 shows a *bc* program for calculation of coefficients for *sin*, as used in *sin_4.c*. Running it with *double* arithmetic in C will produce the same results up through at least 10 digits. Because *bc* uses fixed point arithmetic, it needs extra fractional digits for *sin*, more than are needed for most problems. The same program will work if the *t* function is replaced by *a(x)/x*, with appropriate changes in the interval. The coefficients for log base 2, used in *powf_2.c*, are calculated by having the *t* function evaluate the appropriate Taylor-Maclaurin series. With overnight runs, *bc* can calculate coefficients up to 50 significant digits. These Chebyshev subroutines are adaptations of those given by Press, Flannery et al (1).

Multiple Copy *sin*

Listing 2 shows the four copy *sin* function. The code which performs range reduction, by subtracting off the nearest multiple of *pi*, uses a *rint* function, but takes advantage of the fact that dividing by *pi* does not change the sign. It assumes that addition is performed in the highest available precision, which may be more than *double*. *rint* is not covered by standards, and its result may depend on rounding mode, so it would not take care of portability. Use of *long double* precision in these operations is highly desirable, but of little value unless a true *long double* value of *pi* is available. *long double* should prevent degradation of accuracy for arguments up to $pi * 10^{(LDBL_DIG - DBL_DIG)}$.

The sign of the argument is *ored* into the rounding constant in order not to tie up as many *double* registers, so that the operations on subsequent copies will not be delayed. This procedure avoids branching on processors which do not have a select operation.

Portability at the expense of speed can be obtained using expressions such as

```
pm = (int)(x/pi+(x>0?.5:-.5))
```

or

```
pm = (int)(x/pi-.5+(x>0))
```

since, if *fabs(x/pi)* exceeds *INT_MAX*, there probably aren't more than three digits significance left. Since FORTRAN and Pascal have *round double* to integer syntax, certain processors (e.g. MIPS) have implemented it as a single instruction, which is not used by C compilers.

The integer overflow situation is reported as *errno=ERANGE*, without distinguishing which of the four arguments caused it. Non-portable code for testing the exponent field to identify this situation is used because, on the system where the code was tested, there weren't enough *double* registers to squeeze in any more standard arithmetic without stretching the code out by 30%. There are ways to test whether *pm* is odd without ever casting to *int*, so that range errors are avoided out to *pi/DBL_EPSILON*, but it's not worth the trouble.

Covering the whole interval from $-pi/2$ to $pi/2$ with a single curve fit avoids conditional branches which are particularly troublesome for vector or vector chunk coding. An eight term Chebyshev-economized polynomial is just sufficient to hold the errors to 1 unit-in-the-last-place with *DBL_MANT_DIG* = 53, in the absence of other approximations.



© 1992 AT&T

BUSTER.

Think how much more productive your development process could be if you could minimize the time you put into the administration of tests.

AT&T has done just that with a test administration and execution system called BUSTER.

BUSTER allows tests to be executed without human intervention, reducing system test intervals and staffing levels. It's so easy to use, even new or occasional users can handle BUSTER with minimal training, which also leads to shorter testing intervals.

And BUSTER provides standardized timely feedback through the generation of selected reports. It also promotes test re-use by making tests easier to understand, maintain and share.

If you're looking for proven performance and reliability in your software, call 1 800 462-8146 for our free catalog.

THE INTELLECTUAL PROPERTY DIVISION
Innovation you can depend on.



Listing 3 (sin_4~bt.c)

```
/* Tests sin_4() */
typedef struct {
    double X1, X2, X3, X4;
} ARG_D_4; /* vector 4 */
ARG_D_4 sin_4();
#include <math.h>
main(){
    ARG_D_4 res;
    res=sin_4(-2.,-1.,1.,2.);
    printf(
"\t%.17g\t%.17g\n\t%.17g\t%.17g\n\t%.17g\t%.17g\n\t%.17g\t%.17g\n",
        res.X1,sin(-2.),
        res.X2,sin(-1.),
        res.X3,sin(1.),
        res.X4,sin(2.));
}
/* End of File */
```

C-INDEX/II™

database library

* "terrific"

Wayne Ratliff, author of dBase

* "Its ease of use is remarkable."

C Gazette

windows 3.0®

ms dos®

microsoft c®

turbo c®

unix®

os/2®

borland c++®

metaware high c®

9 easy functions add fast data access to your C programs.

C-Index is the powerful and flexible database software that's also easy to use. For 7 years, programmers have proven it, using C-Index to create award-winning software . . . Experts like Wayne Ratliff (*dBase/Emerald Bay™*), Jack Waller (*Primetime™*), and Jeff Gold (*BusinessWorks PC™*).

Power

- Fast B+Tree Indexing
- Unlimited Key Types
- Unlimited File Size
- Multi-and Single-User Access
- Data and Indexes in Same File
- Multiple Record Formats per File

Ease-of-Use

- 9 Simple High-Level Functions
- Over 40 Additional Functions
- No Application Royalties
- Complete Portable Source Code
- 230-page Manual
- Interactive Example Programs

* "Of the dedicated B-tree systems, the C-Index/II product proved to be the easiest to work with in terms of ease of learning, library functionality, product capabilities, and licensing terms. It is a solid product that will provide B-tree capabilities to any programmer, and is easily ported to virtually any environment." UNIX REVIEW

C-INDEX/II: \$695.00

Advanced database management
for C language applications

C-INDEX/PC: \$195.00

Includes most C-Index/II features,
and is portable to any PC system.

Request your free demo kit today. Call 818/584-9706 or Fax 818/584-0364

Trio Systems

936 E. Green St., Suite 105, Pasadena, CA 91106 U.S.A.
Telephone 818/584-9706 • Facsimile 818/584-0364

◆ Request 169 on Reader Service Card ◆

Putting the interval end points where the function has zero slope helps prevent round off error from introducing discontinuities.

Horner polynomial evaluations are begun before the sign of the result has been determined, leaving the sign switching to be performed when the compiler finds the necessary pipeline slots. The third and fourth polynomial evaluations will lag well behind the first and second, so the third and fourth Horner polynomials are split in two so that the pipelines can be kept fuller after the earlier polynomial evaluations are complete. This adds two multiplications and one possibly significant round off error in each of the third and fourth results.

The fourth copy differs from the third only in that the code is written with parentheses to force the final additions to occur in the most parallel (but not most accurate) order. The dummy multiply by 1 is needed to force K&R compilers to honor the parentheses, but has no effect in ANSI syntax. Since similar techniques are used to a greater extent in scalar math libraries for super-scalar processors, these less accurate results are likely to be closer to the scalar results.

This function should achieve a megaflop rating better than the LINPACK rating on many processors, which is unusual effectiveness for such complex code. One of the ways it could be used would be to combine calculation of unrelated *sins* and *cos*s, using the relationship

```
#define cos(x) sin(PI/2-(x))
```

as needed. A similar tactic should pay off on vector architectures, in which the various arguments are copied to a temporary vector so that the vector *sin* function can be used.

Effective pipelining of this function appears to require more than 16 double registers, along with special efforts to perform as many calculations as possible in *int* registers. Examination of results of an early MIPS compiler showed that it was able to economize on the size of generated code by setting the constants only once. Like many RISC architectures, MIPS has immediate constants available only to initialize registers, not to participate directly in

floating point operations. This may not leave enough registers available for extensive pipelining.

Optimization for reduction of length of generated code prior to scheduling of operations is less well correlated with execution speed on pipelined than on scalar processors. The MIPS software does not report the number of empty pipeline stages. The compiler for the original Multiflow 7/200 compiles this code in 96 major instruction cycles and obtains a superscalar speedup factor of 4. Only six of these instructions are empty, all occurring after the first copy result is complete. *sin_4* on the Multiflow is twice as fast as their library *sin*, giving four results in 12 microseconds. On the Silicon Graphics 4D/25, both *sin_4* and the library *sin* take about four microseconds per result.

Listing 3 shows a test driver to compare the results of *sin_4* with *sin*. While many compilers allow passing a *double* to a function which receives it as a union, other compilers push a union on the stack in a different order from a plain *double*. It is safer to make *sin_4* copy the arguments into its unions. On one of the compilers tested, the generated code is the same either way.

The Chebyshev fit of Listing 1 can be changed to use *sin_4*, after changing from *bc* to C syntax. The order of Chebyshev fit may as well be a multiple of 4. The accuracy of math function approximations, such as the functions discussed in this article, can be tested by fitting Chebyshev polynomials and comparing the coefficients with those obtained by a higher accuracy calculation in the same interval.

Multiple Copy Float *cos* and *sin*

Listing 4 shows a function to calculate *cos* and *sin* of two arguments in *float* precision. Since it uses rational polynomial approximations, there is more built-in opportunity for parallelism than in a Horner polynomial, and two such functions are enough to fill a four stage pipeline at the peak stages. Without prototypes, the only way to pass *float* arguments without widening to *double* is by unions. With prototyping, it would be better to pass *float* arguments and copy them to unions inside the function.

One multiply can be eliminated from the critical path by scaling the arguments to multiples of $\pi/2$ and adjusting the polynomial coefficients accordingly. The division by 2 of the half-angle formulae is buried in the coefficients, so the range reduction maps the arguments into the range -2 to +2. Adding and subtracting $4/LDBL_EPSILON$ produces a number which is rounded to the nearest multiple of 4. As long as promotion to IEEE *double* is used, so that no precautions against underflow are needed, there would be no problem in changing the scaling so that the code could start off

```
tn = x1.flr/2/PI - rint(x1.flr/2/PI)
```

in case that could be calculated more efficiently. The choice of scale was influenced by the desire to maintain accuracy if base 16 arithmetic is used.

Scaling the arguments would produce an additional round off error if the calculations were performed in *float* precision, but *double* is almost mandatory anyway as it prevents degradation of accuracy for arguments

up to $2\pi \cdot 10^{(DBL_DIG-FLT_DIG)}$. A warning such as storing a value into *errno* could be provided when larger arguments arrive, but this is not clearly a failure meriting the *ERANGE* label unless the argument becomes so large that the *rint* code won't work. Basing the *errno* calculation on values which are calculated anyway minimizes the use of additional registers.

The first rational polynomial is calculated Horner style, and the last attempts to catch up by calculating all terms individually, at the cost of one additional multiplication. The scheme of eliminating one of the coefficients by choice of scale allows the numerator to get a head start so that the final multiplication can be performed without delaying the division. The compiler may have to be forced into performing the first add in the denominator without waiting until the last term has been calculated. Certain compilers insist on converting the repeated divisions into multiplications, which is no problem when the operations have been promoted in precision.



CALICO™

© 1992 AT&T

When you select a software development environment, power and performance are always a top priority.

That's why AT&T has introduced CALICO, an object-oriented programming language supported by a complete stable of integrated development tools.

CALICO speeds the development process by incorporating the best features of all OOP languages: a simple, uniformly-applied object model, a new multiple inheritance model, a large, powerful library of objects, a graphical tool-builder, a natural interface to C and companion to C and C++, plus effective support for the team development process. All of which allow you to focus on applications rather than programming.

If you're looking for proven performance and reliability in your software, call 1 800 462-8146 for our free catalog.

THE INTELLECTUAL PROPERTY DIVISION

Innovation you can depend on.



Vector Chunk *float pow*

The *pow* function in C is expected to embody two entirely different types of operation. In order for it to be vectorizable, or to obtain good vector chunk performance enhancement with current compilers, it has to be restricted to the cases of positive base, where it can be replaced in effect by

```
#define pow(x,y) exp(log(x)*y)
```

This could be done with a top level *powf_2* which determines whether both pairs of arguments are of one type, and, if so, invokes an appropriate vector chunk function. The usual test is whether *y1* and *y2* are changed by casting to *int* and back to *float*. It doesn't hurt much to use the *log* treatment anyway, unless *x* is negative. If the argument pairs cannot be processed by the same algorithm, it would have been more efficient not to have tried to treat them as a vector chunk at all.

The function of Listing 4 does not take care of the negative base case, which is OK according to ANSI standards if it is called as the implementation of the FORTRAN real exponentiation operator. I use it in this form in time marching aerodynamics codes, where it gets executed millions of times.

Promotion to *double* is really needed only in the sections involving addition of the integer exponent to the base range *log2* up to the splitting of the *exp2* argument into an integer plus or minus a fraction, and then only when the result is far from 1. The somewhat complicated system for splitting the base into modified *frexp* form works quickly and accurately

IF YOUR PROGRAM IS TOO BIG OR TOO SLOW AND THE THOUGHT OF DESIGNING OVERLAYS MAKES YOU WANT TO THROW UP AND YOU'VE GOT A DEADLINE TO MEET AND YOU USE PLink86+, RTLink+ or RTLink+ AND ALL YOU WANT IS SOMETHING THAT'LL MAKE YOUR OVERLAYS

SMALLER AND FASTER WITHOUT A LOT OF HYPE OR HASSLE, DROP EVERYTHING AND

CALL (303) 938-1210. It'll connect you to PEOPLE WHO REALLY

KNOW OVERLAYS, who WROTE OVERLAY ARCHITECT+ & OVERLAY OPTIMIZER+ (OA/OO;

FORMERLY "LTO"), and who can tell you how OA and OO

overlay data and handle indirect calls automatically, AND

possible in a size you specify (often smaller and faster than VMLI). **HOW MUCH** for all this

technology? gets you BOTH programs, a 30-day MBG, friendly technical support,

and we take PLASTIC. Whew!

ATLAST!
SOFTWARE INC.

© 1992 Atlast Software, Inc. Overlay Architect and Overlay Optimizer are trademarks of Atlast Software, Inc. All other trademarks are property of their respective owners.

900 Baseline Road
Ottawa, A-1
Burlington, CO 80302

Listing 4 (powf_2.c)

```
typedef struct {
    float x1, x2;
} ARG_F_2; /* vector 2 */
#include "float.h"
#include "errno.h"
#ifdef errno
extern int errno;
#endif
#define MANTBITS (FLT_MANT_DIG - 1)
ARG_F_2 powf_2(x1, y1, x2, y2)
union fltfmt {
    float flt;
    int iflt; /* VAX: must change all this */
    struct ffmt {
        unsigned int exp;
        unsigned int mant:MANTBITS;
    } ffmt;
} x1, x2, y1, y2;
{
    #define max(i,j) ((i)>(j)?(i):(j))
    #define min(i,j) ((i)<(j)?(i):(j))
    #if FLT_MANT_DIG != 24
    #error "use portable frexp() ldexp() */
    #endif
    #if FLT_ROUNDS == 1
    #if defined(_STDC_)
    /* This works on some non-ANSI compilers */
    #define ROUND(x) ((x)>=0? \
        (x)+1/LDBL_EPSILON)-1/LDBL_EPSILON: \
        ((x)-1/LDBL_EPSILON)+1/LDBL_EPSILON)
    #else
    #define ROUND(x) ((x)>=0? \
        (x)+1/LDBL_EPSILON)-1/LDBL_EPSILON: \
        ((x)-1/LDBL_EPSILON)+1/LDBL_EPSILON)
    #endif
    #else
    #define ROUND(x) ((x)>=0?(int)(x*.5):(int)(x-.5))
    #endif
    int m1, m2, msign;
    double xr, x2, r, r1;
    ARG_F_2 res;
    /* Copy 1 */
    /* This frexp() operation would be done better after
    ** promotion to double
    ** but it's not mandatory unless dealing with gradual
    ** underflow;
    ** it would eliminate most cases of 0 and inf changing
    ** to finite numbers
    ** if((x1.flt==frexp(x1.flt,&m1))<sqrt(.5)){
    --m1;
    x1.flt *= 2;
    } */
    m1 = ((x1.iflt & 0x7fffffff) -
        (m2 = (x1.fmt.mant < 0x3504f3 ?
            (2 - FLT_MIN_EXP) << MANTBITS :
            (1 - FLT_MIN_EXP) << MANTBITS)))
    >> MANTBITS;
    if (x1.iflt < 0 || x2.iflt < 0) errno = EDOM;
    x1.iflt = m2 | x1.fmt.mant;
    /* Mult by y distributed to increase parallelism */
    r1 = (xr = (x1.flt - 1) / (x1.flt + 1)) * y1.flt;
    x2 = xr * xr;
    /* Coefficients determined by Chebyshev fitting
    ** double precision is only really needed from here */
    r = y1.flt * ((double) m1 + r1 * (2.8853904 +
        x2 * (.5958 * x2 + .961588)));
    /* Msign = (r - rint(r)) < 0 */
    msign = (r - r1 = ROUND(r)) < 0;
    r = 125.0718418 + (x2 - r * r);
    x2 = 360.8810526 + 17.3342336 * x2;
    /* X1.flt = ldexp((x2+r)/(x2-r),(int)r1) */
    x1.flt = (x2 + r) / (x2 - r);
    /* Preferably do this ldexp() operation in double,
    ** but it's slower,
    ** even though msign can be eliminated;
    ** it would always give inf rather than NaN
    ** and would allow use of gradual underflow */
    x1.iflt += (max(FLT_MIN_EXP - 2 + msign,
        min(FLT_MAX_EXP + msign, (int) r1)) << MANTBITS);
    /* X.fmt.exp+=m1; with limiting to prevent exponent wraparound */
    res.x1 = x1.flt;
    /* Copy 2 */
    m1 = ((x2.iflt & 0x7fffffff) -
        (m2 = (x2.fmt.mant < 0x3504f3 ?
            (2 - FLT_MIN_EXP) << MANTBITS :
            (1 - FLT_MIN_EXP) << MANTBITS)))
    >> MANTBITS;
    x2.iflt = m2 | x2.fmt.mant;
    r1 = (xr = (x2.flt - 1) / (x2.flt + 1)) * y2.flt;
    r1 = x2 * xr * xr;
    r = y2.flt * ((double) m1 + xr * (2.8853904 +
        r1 * (.5958 * x2 + .961588)));
    msign = (r - r1 = ROUND(r)) < 0;
    r = 125.0718418 + (x2 - r * r);
    x2 = 360.8810526 + 17.3342336 * x2;
    x2.flt = (x2 + r) / (x2 - r);
    x2.iflt += (max(FLT_MIN_EXP - 2 + msign,
        min(FLT_MAX_EXP + msign, (int) r1)) << MANTBITS);
    res.x2 = x2.flt;
    return res;
}
/* End of File */
```


without widening on a system without gradual underflow. On architectures such as VAX which use a different byte order for *float* and *int*, the unions and constants are different.

If gradual underflow is to be supported without widening the precision, it will require special case treatment. To reduce degradation of accuracy if widening is not used for addition of the integer and fraction parts of the *log2* function, *log2* should be split into a power of 2 plus a smaller term. This leads to complicated code which may require branching, thus defeating attempts to gain pipelined performance.

Evaluation of $\log_2(x_2) * y_2$ is speeded up by grouping the terms in pairs. The calculation $\log_2(x_2) * y_1$ then becomes a bottleneck until the multiplication by *y1* is distributed onto the two groups, one of which consists of the three-term Horner polynomial. Multiplication of *y1* by the integer exponent is performed well before it is needed.

Making such detailed adjustments for a given system is possible only with readable assembly language which displays the final scheduling of the pipelined operations, and is helped greatly by static profiling which gives the effective clock count for each block of generated code. Since we try to write these functions so that there is only one code block, and there are few memory accesses which could introduce bus delays, the speed will not depend on data and there should be no question what effect each change has on speed.

In order to make the *ROUND* macro work the same under K&R syntax as it would in ANSI C, dummy multiplications by 1 are introduced. Otherwise it is a matter of luck whether a K&R compiler will generate the required code, although the left associativity of the + and - operators should produce a preference for left to right evaluation. From an algebraic point of view, *ROUND* would do nothing, and AI techniques could conceivably allow a compiler to know this. The peculiar syntax of K&R which requires such multiplications by 1 makes it semi-obligatory for optimizing compilers to eliminate the redundant operation, unless compiling for an architecture which may generate faster code with alternating multiplication and addition.

If the compiler is unable to generate efficient code for the *max* and *min* macros, it would be better to perform the *ldexp* operations on *doubles* and hope that the extra range of *double* will take care of over and underflows.

Multiple Copy *tan*

The *tan_2* function (Listing 5) requires the least non-portable coding for optimum results, but it illustrates optimizations which have not appeared in the functions discussed above.

Range reduction consists simply of subtracting the nearest multiple of π , and there is no advantage in playing games with unions. The comparisons

start into the pipeline first and are completed before the divides, which may have been converted to multiplications by the compiler.

The remainder of the calculation consists of evaluation of a rational polynomial. On a machine with a divide which pipelines at the same intervals as the other operations, straightforward Horner evaluation of the numerators and denominators might work as well as anything. On the processor for which this code was tuned, a divide operation delays the pipeline, but does not affect addition. For this reason, the order of operations is set up to push all of the multiplications into the pipeline before the divides, as well as to start the first division as soon as possible.

The terms of the numerator and denominator are grouped so that operations are always ready to start into the pipeline, and to take advantage of architectures which have separate pipelines for multiplication and addition.

In the denominator of the first copy, the high order terms must be added in order of increasing complexity. This could be done with parentheses with an ANSI compiler. Possibly better accuracy could be obtained by adding the high order term last. Order is not dictated in the low order terms of the first copy, so as to avoid delaying the second copy. In the numerator of the second copy, the final multiplications are distributed so that they may be performed before the final addition, in order to get the multiplications out of the way early, as well as to allow the second calculation to begin to catch up with the first.



© 1992 AT&T

SABLIME™

Pulling all the pieces together in your software development process and tracking every release is easy when you've got SABLIME.

SABLIME tracks changes to a product's software, hardware, firmware and documents – from origination through maintenance and delivery to support – creating a united development environment which increases productivity and protects your investment in programmer training.

This versatile system features extensive reporting and query capabilities, in-process data collection, quality assurance, data generation, distributed environment operation, intra- and inter-project communications, control flexibility and user-friendly interface choices.

If you're looking for proven performance and reliability in your software, call 1 800 462-8146 for our free catalog.

THE INTELLECTUAL PROPERTY DIVISION

Innovation you can depend on.



Listing 5 (cosf_~bs.c)

```
typedef struct {
    float cos1, sin1, cos2, sin2;
} ARG_FF_2; /* vector 2 pairs */
ARG_FF_2 cosf_sinf_2(x1, x2)
{
    union {
        float flt;
        int iflt;
    } x1, x2;
} /* 2 pair single precision
sin/cos function */

#include "float.h"
#if FLT_ROUNDS != 1
    #error "rounding mode not nearest, fix code"
#endif
#include <errno.h>
#ifndef errno
    extern int errno;
#endif
#include <math.h>
#define M_2_PI 0.63661977236758134308
#define T2PI 2*M_2_PI
#define TP2 T2PI*T2PI
#define TP3 TP2*T2PI
#define TP4 TP2*TP2
#define TP5 TP3*TP2
ARG_FF_2 res;
double tn, td, r;
/* integer compare with 0 same as float, for IEEE
** since arg comes in int register, this is faster
**
** doing everything in double, we won't lose accuracy
** by converting arg to multiple of PI/2
** this allows range reduction by subtracting an integer
**
** reduce to range +- 2, divide by 2 later
** when it cannot underflow */
tn = x1.flt * M_2_PI + (td = x1.iflt >= 0 ?
    -4 / LDBL_EPSILON : 4 / LDBL_EPSILON);
if (fabs(x1.flt * M_2_PI) >= 4 / LDBL_EPSILON |
    fabs(x2.flt * M_2_PI) >= 4 / LDBL_EPSILON)
    errno = ERANGE;
tn -= td;
tn = x1.flt * M_2_PI - tn;
td = tn * tn;
/* divide arg by 2 and rationalize numerator and denominator
** numerator of rational approx for tan(x1/2)
** Horner polynomials 1st time */
tn = 886.77348 * TP4 + td * (-99.398954 * TP2 + td);
/* denominator */
td = 886.77346 * TP5 + td *
    (-394.98971 * TP3 + td * 14.425694 * T2PI);
/* cos, sin half angle formulae, rationalized */
res.cos1 = (td * td - tn * tn) / (td * td + tn * tn);
res.sin1 = (tn * td + tn * td) / (td * td + tn * tn);
/* copy 2 */
tn = x2.flt * M_2_PI + (td = x2.iflt >= 0 ?
    -4 / LDBL_EPSILON : 4 / LDBL_EPSILON);
tn -= td;
tn = x2.flt * M_2_PI - tn;
td = tn * tn;
/* distribute terms to finish polynomials quicker */
tn = 886.77348 * TP4 - td * 99.398954 * TP2 + td * td;
r = 886.77346 * TP5 - td * 394.98971 * TP3;
td = r + td * td * 14.425694 * T2PI;
res.cos2 = (td * td - tn * tn) / (td * td + tn * tn);
res.sin2 = (tn * td + tn * td) / (td * td + tn * tn);
return res;
}

/* End of File */
```

Listing 6 (tan_2.c)

```
typedef struct {
    double X1, X2;
} ARG_D_2; /* vector 2 */
ARG_D_2 tan_2(x1, x2)
{
    double x2, x, n1, x4;
    ARG_D_2 res;
#include "float.h"
#if FLT_ROUNDS != 1
    #error "rounding mode not nearest; adjust code"
#endif
#if FLT_RADIX != 2 && FLT_RADIX != 10
    #error "code not optimum for accuracy in this
RADIX"
#endif
#include <errno.h>
#ifndef errno
    extern int errno;
#endif
#include <math.h>
#define M_PI 3.14159265358979323846
x2 = (n1 = (x1 > 0 ? 1 / LDBL_EPSILON :
    -1 / LDBL_EPSILON)) + (x = x1) / M_PI;
x -= (x2 - n1) * M_PI;
if (fabs(x1 / M_PI) >= 1 / LDBL_EPSILON |
    fabs(x2 / M_PI) >= 1 / LDBL_EPSILON)
    errno = ERANGE;
/* now in 1st or 4th quadrant */
#define c0 33281881.3202530279
n1 = c0 + (x2 = x * x) * (-15666569.8711211851);
x4 = x2 * x2;
res.X1 = x * (c0 + x2 * (-4572609.43103684572) + x4 *
    (131095.887915363619 + x2 * (-968.863245687503149 +
    x2))) / (n1 + x4 * (915701.668921990803
    + x2 * (-13491.7937027796916)
    + x4 * 44.4083322286368691));
/* copy 2 */
x2 = (n1 = (x2 > 0 ? 1 / LDBL_EPSILON :
    -1 / LDBL_EPSILON)) + (x = x2) / M_PI;
x -= (x2 - n1) * M_PI;
n1 = 915701.668921990803 - (x2 = x * x)
    * 13491.7937027796916;
x4 = x2 * x2;
res.X2 = (x * (c0 + x2 * (-4572609.43103684572)) +
    (131095.887915363619 + x2 * (-968.863245687503149 +
    x2)) * x4 * x) / (c0 + x2 * (-15666569.8711211851) +
    x4 * (n1 + x4 * 44.4083322286368691));
return res;
}

/* End of File */
```


Summary

Much of this article will appeal only to those who like to tweak code for another 20% in performance. I have concentrated on the points where architectural dependencies pop up and tried to show where their impact can be reduced for relatively small performance penalties. Math library functions are probably the closest thing to applications where non-portable code is appropriate. This is the reason for these functions (at least the scalar versions) being defined in the C standard so that they need not be carried as part of an application.

The extent to which special vector chunk functions should be used to perform math library operations in groups may be questioned. An application which uses these functions probably should provide an alternative header file which will cause them to be replaced with standard functions. Compilers are most likely to begin to incorporate such functions automatically if they produce benefits on the standard benchmarks.

Scalar math functions can detract from the performance of superscalar processors. The techniques shown enable superscalar performance to be obtained in the evaluation of grouped math library functions. In typical applications, the percentage of execution time spent in math functions can be reduced in comparison with a scalar processor. □

References

John Ellis, *Bulldog: A Compiler for VLIW Architectures*, MIT Press, 1985 (avoidance of code explosion, trace scheduled pipelined code).

Press, Flannery, Teukolsky, Vetterling, *Numerical Recipes in C*, Cambridge, 1988 for Chebyshev analysis.

John Palmer, Stephen Morse, *The 8087 Primer*, Wiley, 1984 for some math function concepts.

P. J. Plauger, Jim Brodie, *Standard C*, Microsoft Press, 1989 for the simplest satisfactory explanation of *float.h*, *limits.h*, *math.h*.

T. Prince, "Generating Source for <float.h>," *The C Users Journal*, V8N6, June 1990.

NEW WINDOWS PRODUCT

SafeWin Quality Assurance DLL

- Automatically verifies the integrity of your windows applications.
- No recompilation necessary. Just relink.
- Saves hours of difficult windows debugging.
- Monitors functions from all the major API categories (including: atom, GDI, memory, module, resource and segment management functions) and automatically reports failures.

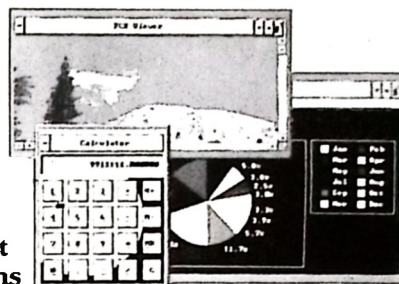
Supports: Borland C++, Microsoft C, Windows 3.0.

USE 4 SIGHT NOT HINDSIGHT

4SIGHT

Application Framework

- Text • Graphics • C LIB
- C++ CLASSLIB • Bitmaps • Icons
- Help Manager • Event Manager
- Resource Manager • DOS
- 16/32 Bit Extended DOS
- Extensible • CUA Compliant
- Over 100 Sample Programs



Supports: Borland C++, Microsoft C, WATCOM C, WATCOM C/386, TopSpeed C, TopSpeed C++, Zortech C++, Intel C Code Builder, MetaWare High C 386, PharLap 286, PharLap 386, DOS4GW, BGI and MS Graphics, Genus GX Graphics, MetaWindows.

HEAP SEEKING SOFTWARE

MoreHeap Heap Manager/ Virtual Memory Manager

- Includes replacement for malloc() that recognizes QEMM, HIMEM.SYS, and DOS 5, adding up to 348 K of memory transparently—No reprogramming for this feature just relink.
- Reduces heap fragmentation.
- Low overhead, ideal for lots of small allocations.
- Handle-based virtual memory manager adds up to 128MB of swappable, discardable, and moveable memory.
- LRU paging overflows to expanded memory, extended memory and disk.
- Detects common heap errors like unfreed blocks, doubly freed pointers, and buffer overwrites.

SafeHeap Memory Debugger

- Just relink to include SafeHeap's validating memxxx() and strxxx() routines—rerun your application to produce a report of suspected buffer manipulation problems. No recompilation.
- Documented interface for writing your own validating routines.
- Small library.
- Reports invalid reads/writes to near/far heap, globals, and autos, including name of calling function, affected global variables and heap areas, and additional call statistics.
- Zero overhead per allocation, uses HMA for storage.
- Supports all memory models.

MoreHeap & SafeHeap Support: Borland C++ and Microsoft C.

Call for free demo disks and literature.

Visa/MasterCard/COD/POs accepted.
Shipping and handling additional.

MoreHeap	\$135	SafeWin	\$149
SafeHeap	\$ 95	4Sight	\$199 and Up
MoreHeap/SafeHeap	\$179		



SeaBreeze Software Systems
1330 Highway 206, Ste. 114
Skillman, NJ 08558
Sales/Support: 609 924 6793
BBS: 609 683 7701
CIS: 72330, 705

Porting Command Line

William Smith

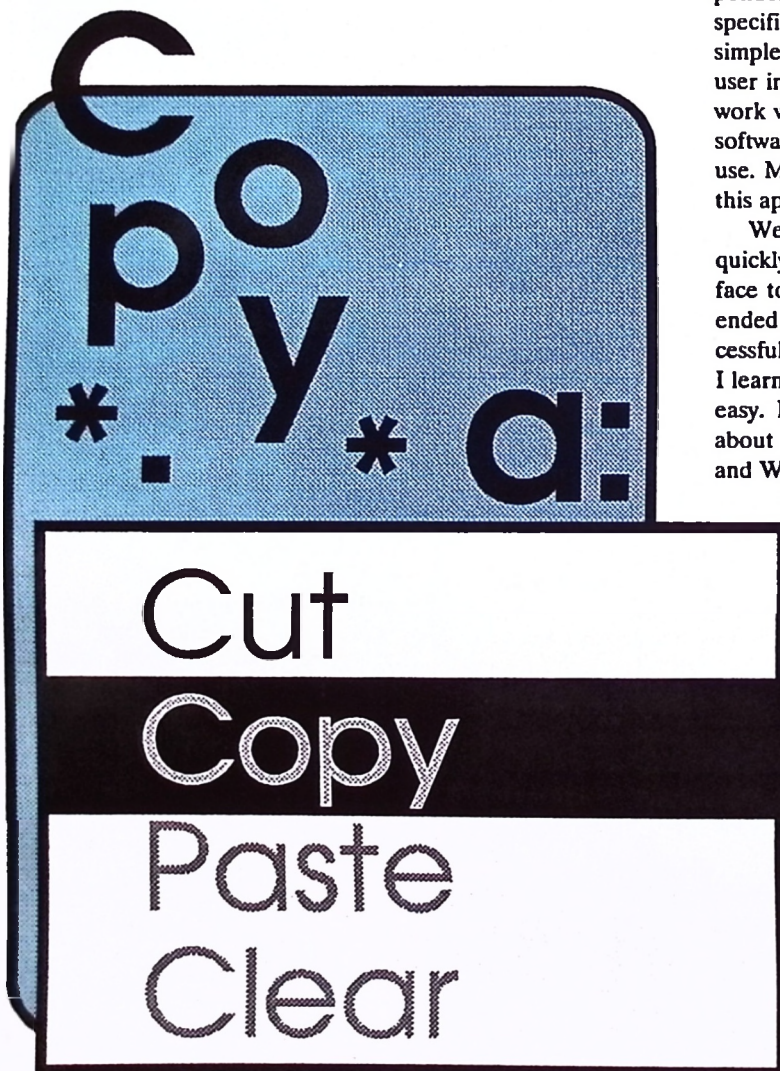
Not very long ago the computer industry's direction concerning operating systems and user interfaces was uncertain and confusing. You had to be psychic, lucky, or just plain good at reading the writing on the wall to assess the direction the industry was going. To name just a few of the possibilities, there was UNIX and X-Window, OS/2 and Presentation Manager, MS-DOS and a primitive Windows 2.0, and MS-DOS and countless lesser known third party user interface libraries. Right at the height of this confusion, I was faced with choosing an operating system and user interface for a software development project. I was developing a data acquisition and data management system for an automated athletic weight

training system. The customer also wanted the program to have a modern graphical user interface (GUI).

For economic reasons and availability of development tools, I decided to write the software in C for a target 80386 personal computer running MS-DOS. The user interface decision was much harder. I, like many people when faced with a tough decision, decided not to decide. I made an engineering decision to do the user interface last. This allowed me to postpone the commitment to a specific user interface library. This was contrary to many opinions about designing software at the time. Instead of designing the software from the user interface and screen level first, I designed the software by identifying core functionality and operations independent of the user interface. To get the project going, I specified that the software was to be first developed using a simple command line interface (CLI). Later, when a graphical user interface library was chosen, I would port the program to work with the GUI library. As soon as a clear direction in the software industry emerged, I would decide upon what GUI to use. Much to my delight and the satisfaction of the customer, this approach eventually paid off.

Well as most of you know, Windows 3.0 came out and quickly became a success. It became obvious what user interface to use. Eventually the project was over and the customer ended up with both a command line version and after a successful port, a Windows version of the software. Along the way I learned a lot about the approach involved in making this port easy. I am going to share with you some of what I learned about porting command line interfaces to GUI's in general and Windows in particular.

The concept of porting I am going to discuss requires work. It is not a simple recompile under a different environment. Granted, there are some features built into Microsoft QuickC for Windows



William Smith is the engineering manager at Montana Software, a software development company specializing in custom applications for MS-DOS and Windows. You may contact him by mail at P.O. Box 663, Bozeman, MT 59771-0663.

User Interfaces to GUIs

and version 3.0 of Borland C++ that allow you to recompile your code under Windows with no changes. The approach is easy, but all it gets you is a command line within a window. I do not consider this a true GUI. To get true GUI behavior in your program you are going to have to write some code and make some changes. GUIs are here to stay and will become increasingly popular in the future. The effort to port your code to a GUI is worth it. With proper planning, it does not have to be that painful either.

CLI Versus GUI

With the popularity of GUIs, CLIs may seem antiquated, but they have their place. They are easy to write and if you stick to the standard C library, very portable. You can generate a test program for exercising new code quicker using a CLI than a GUI. CLIs are fast and can be easy to use. All that the program requires of the user is to type in the program name and some options at the operating system prompt. The command line interface is elegant in its simplicity and appreciated by the skilled software user. CLIs are also very adaptable to batch mode processing. Unfortunately, CLIs are cryptic and require the user to have knowledge and memory of the command line syntax. By providing a help screen defining proper usage, you can relieve this challenge somewhat for the inexperienced user. An easy to use CLI program should provide a provision to invoke this help screen when an *h* or *?* option is passed on the command line. The program should also display the help information when there is an error in the command line syntax.

GUIs are visually appealing and easier to negotiate than CLIs especially for the unskilled user. GUIs can require more steps to accomplish the same task than a CLI and are not as conducive to batch mode processing as CLIs. With some effort, you can add key stroke short cuts and batch ability to GUIs to satisfy the demands of the advanced user.

Table 1 contains a list of user interface characteristics and features. It rates CLIs and GUIs for comparison.

CLI Translated to GUI

Programs in the simplest terms require input, perform some task and generate output. The input information is in the form of instructions and data. The output information consists of results and data. With a command line interface your choices of how to communicate instructions to a program are limited. Typically, CLIs use single characters, sometimes preceded by a delimiting character such as */*, to specify options, commands, or flags. The user passes data to CLIs in the form of file names or lists of strings. On the other hand there are many more options available on how to communicate information to a program that employs a GUI. Table 2, lists the basic command line interface elements and corresponding GUI elements. The nomenclature is based on Windows. Notice there is not a one-to-one correspondence between a command line element and a GUI element. With a GUI, there can be many different ways to accomplish the same task. This gives the developer some flexibility in designing the interface.

Listing 1 contains a code fragment from the *main* function of a program that processes a command line and performs database operations. This is an excerpt from a data base utility program that I first created as a CLI program and later ported to Windows. The command line is simple. To specify an operation, the program requires a single character as the first argument on the command line. The second and third arguments are a file name and a key name. The operation chosen determines which of these last two arguments are required.

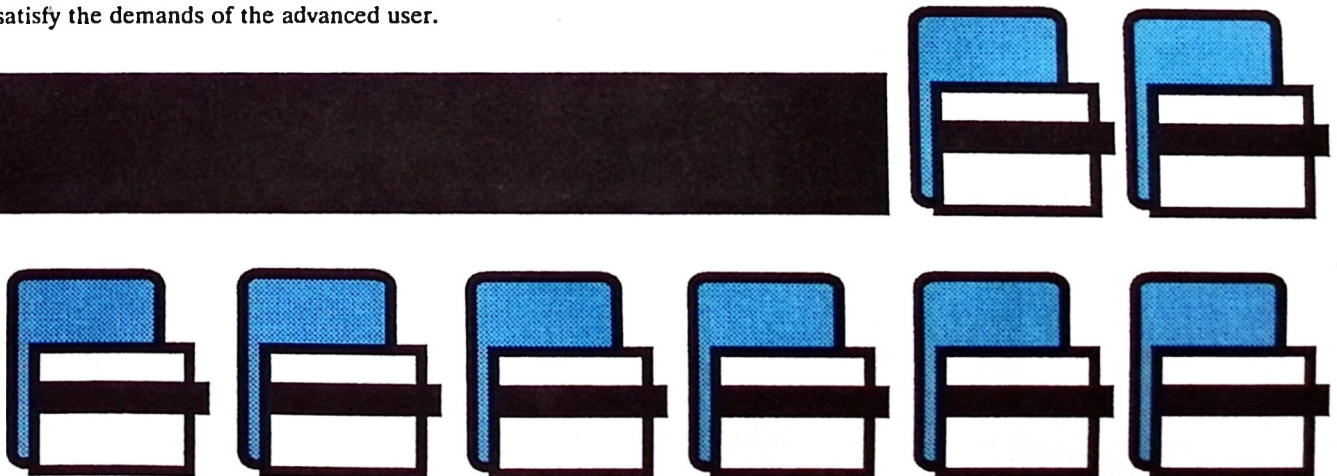


Table 1

Characteristics / Features	CLI	GUI
Ease of Development	Good	Poor
Portability	Good	Poor
Batch Processing	Good	Poor
Ease of Use	Poor	Good
Aesthetics and Presentation	Poor	Good
Visibility of Information	Poor	Good
Flexibility of Program Operation	Poor	Good
Number of Interaction Steps	Few	Many

User Interface Issues, CLI Versus GUI

Table 2

CLI Element	Corresponding GUI Elements
Option / Action Flag	Menu Item, Radio Button, Check Box
String Unlimited Choices	Edit Box, Check Box
String Limited Choices	Menu Item, Radio Button, List Box
File Name	Edit Box, List Box, Dialog Box
Script File	Editor, Dialog Box, Custom Builder

CLI Elements and Analogous GUI Elements

Listing 1

```

switch ( argv[1][0] )
{
    case 'a':
    case 'A':
        status = add_data_to_db( argv[2] );
        break;
    case 'd':
    case 'D':
        status = del_data_from_db( argv[2] );
        break;
    case 'g':
    case 'G':
        status = get_data_from_db( argv[2], argv[3] );
        break;
    case 'l':
    case 'L':
        status = list_keys_in_db();
        break;
    case 'r':
    case 'R':
        status = replace_data_in_db( argv[2], argv[3] );
        break;
    case 'v':
    case 'V':
        status = vrfy_data_in_db( argv[2], argv[3] );
        break;
    default:
        status = FAIL;
        break;
} /* switch ( argv[1][0] ) */

/* End of File */

```

Listing 2

```

switch ( wParam )
{
    case IDM_ADD:
        /* Select a File */
        status = FileSelectDialog( hInstance, hWnd,
            Caption, FileSpec );
        if ( status == FAIL || status == FALSE )
            break; /* Canceled the operation. */
        /* Add file to database */
        status = add_data_to_db( FileSpec );
        break;
    case IDM_DELETE:
        /* Select a Key */
        status = KeySelectDialog( hInstance, hWnd,
            Caption, Key );
        if ( status == FAIL || status == FALSE )
            break; /* Canceled the operation. */
        /* Delete Keyed Data from database */
        status = del_data_from_db( Key );
        break;
    case IDM_GET:
        /* Select a Key */
        KeySelectDialog( hInstance, hWnd,
            Caption, Key );
        if ( status == FAIL || status == FALSE )
            break; /* Canceled the operation. */
        /* Select a File */
        FileSelectDialog( hInstance, hWnd,
            Caption, FileSpec );
        if ( status == FAIL || status == FALSE )
            break; /* Canceled the operation. */
        /* Get data from database and store in File */
        status = get_data_from_db( Key, FileSpec );
        break;
    case IDM_REPLACE:
        /* Select a Key */
        KeySelectDialog( hInstance, hWnd,
            Caption, Key );
        if ( status == FAIL || status == FALSE )
            break; /* Canceled the operation. */
        /* Select a data file */
        FileSelectDialog( hInstance, hWnd,
            Caption, FileSpec );
        if ( status == FAIL || status == FALSE )
            break; /* Canceled the operation. */
        /* Replace Key data with data in FileSpec */
        status = replace_data_in_db( Key, FileSpec );
        break;
    case IDM_VERIFY:
        /* Select a Key */
        KeySelectDialog( hInstance, hWnd,
            Caption, Key );
        if ( status == FAIL || status == FALSE )
            break; /* Canceled the operation. */
        /* Select a File */
        FileSelectDialog( hInstance, hWnd,
            Caption, FileSpec );
        if ( status == FAIL || status == FALSE )
            break; /* Canceled the operation. */
        /* Verify Data in database (Key) matches
        ** data in FileSpec */
        status = vrfy_data_in_db( Key, FileSpec );
        break;
    default:
        status = FAIL;
        break;
} /* switch ( wParam ) */

/* End of File */

```


The command line syntax is

PROGRAM OPERATION FILE KEY

The possible operations are shown in Table 3.

Listing 2 contains an excerpt from a Windows program. This code is taken from a program that accomplishes the same tasks as the program that contains Listing 1. The code is from the windows procedure for the main window that responds to messages for the main window. The code fragment contains a switch statement that reacts to menu choices that are in the form of messages communicated from Windows. There is correspondence between the command line options and the menu choices. The user specifies the file and key through interaction with dialog boxes. Notice that the calls to the functions, `add_data_to_db`, `del_data_from_db`, `get_data_from_db`, `replace_data_in_db`, `vrify_data_in_db` are the same for both the CLI version and the GUI version. The code for these function should be portable across user interfaces.

Table 4 lists the CLI arguments and the corresponding GUI elements used to accomplish the same operations.

GUI Portability Strategy

There are three major guidelines that form the foundation of a strategy for portability between user interfaces:

1. Identify high level functionality and data structures
2. Isolate program functionality from user interface code
3. Use standard library and standard types

Planning for user interface portability requires adopting a design philosophy of first identifying high level program functionality and avoiding the specifics of screen design. The idea is to isolate the major data structures and operations that the program supports. If you can wrap a command line interface around the operations that your program performs, you are on the right track. Granted some programs such as word processors do not lend themselves to command line interfaces. Even in this situation, you can isolate individual functions a word processor performs and group them in a utility program with a command line interface.

Save Disk Space PKZIP version 2.0

"PKZIP is the undisputed champion of data compression" - BYTE Magazine

"It [PKZIP] combines blazing decompression times and excellent compression speed" - PC Magazine

PKWARE introduces the next generation of its award winning compression utility, giving even greater performance levels than achieved with previous releases of the software.

Software developers, you can significantly reduce product duplication costs by decreasing the number of disks required to distribute your applications. Call for Distribution License information.

Put Your Executables on a Diet

PKLITE increases your valuable disk space by compressing executable (.EXE and .COM) files by an average of 45%. The operation of PKLITE is transparent, all you will notice is more available disk space!

Software Developers, save disk space and media costs with smaller executables. You can distribute your software in a compressed form with PKLITE Professional. PKLITE Professional also gives you the ability to compress files so that they cannot be expanded, hindering reverse engineering of your programs.

Compression for YOUR Application

The PKWARE Data Compression Library allows you to incorporate data compression technology into your software applications. The application program controls all the input and output of data, allowing data to be compressed or extracted to or from any device or area of memory.

All Purpose Data Compression Algorithm compresses Ascii or Binary data quickly. An adjustable dictionary size allows software to be fine tuned for maximum compression or speed. The routines can be used with most popular languages, such as C, C++, Pascal, Assembly, Basic and Clipper.

PKWARE INC.
The Data Compression Experts

9025 N. Deerwood Drive Brown Deer, WI 53223
(414) 354-8699 Fax (414) 354-8559

PKZIP \$47.00 PKLITE \$46.00 PKLITE Professional \$146.00
PKWARE Data Compression Library \$295.00
Please add \$3.50 S&H per package in the US & Canada. \$5.00 overseas.
Wisconsin residents add 5% state sales tax & applicable county sales tax.
Visa and Mastercard accepted, no COD orders.

Table 3

Operation	Description
A	add data contained in FILE to database
D	delete data specified by KEY from database
G	get data specified by KEY from database and store in FILE
L	list all the keys in the database to standard output
R	replace data in database specified by KEY with data in FILE
V	verify data in database specified by KEY with data in FILE

Once you define the major functionality, make sure you strongly segregate the code you write to implement this functionality from the user interface code. The modules that contain the core operations of your program should be portable and not make any function calls to a user interface library.

To help with portability, use the standard library functions and the standard types. Some of the issues encountered when porting among GUIs and operating systems are sizes of standard types, structure packing, and alignment. The size of some of the standard types will change from one platform to another. An example is the default *int* type. Under some compilers an *int* is 16 bits while with others it is 32 bits. If you do not care what size it is and want to use the native most effi-

Table 4

CLI Arguments	GUI Elements
Option	Options (Menu)
A	Add...
D	Delete...
G	Get...
L	
R	Replace...
V	Verify...
File name	File Select Dialog Box
Key name	Key Select Dialog Box

CLI Arguments and Corresponding GUI Elements

cient size, use just a plain *int*. If you only need 16 bits and want to conserve space in a platform where *int* is 32 bits use *short int*. If you need 32 bits even in a platform where *int* is 16 bits use *long int*. Avoid *typedefing int* and encoding the size in the type such as *int16* or *int32*. Some claim that this increases portability, but I have found the exact opposite to be true. I also recommend using the *size_t* type defined in standard C. *size_t* is defined as an unsigned integer. It is convenient to use variables of type *size_t* as array indexes.

Related to type size is structure packing and alignment. In most situations, compilers align structure members (except *chars*) on boundaries that correspond to the most efficient type size. On a 16-bit system, structure members are aligned on word boundaries. On a 32-bit system, structure members are aligned on *double* word boundaries. Some compilers allow the program to control structure alignment.

Try to avoid dependencies in your code on type size and structure alignment. The *sizeof* operator can help with types and the *offsetof* macro can help with structures. Pay careful attention to third party libraries if you use them. They may have size and structure alignment dependencies that could bite you later.

Buffer sizes and string lengths should be set using manifest constants. For example, the maximum length of a string to hold a file name may change from one system to another. It is far easier to change the definition of a manifest constant in one place than find all places where space for a string is allocated.

Porting to Windows — the Gruesome Details

Since Windows is such a popular GUI, it is worth talking about some of the specific issues encountered when porting existing C code to this environment. As a first step in porting your CLI program to the Windows GUI, you may want to create a user interface shell and spawn the CLI version of your program using the Windows *WinExec* function call. *WinExec* is similar to the *spawn* function family in standard C. This approach will get you up and running, but I found it unacceptable for a finished product. The major drawback is the lack of and the difficulty involved in communicating between MS-DOS and Windows programs.

SoftC Database Library

SoftC is the database library of choice for professional C & C++ developers. It offers unsurpassed speed and flexibility along with access to industry standard files at a very competitive price.

- Full compatibility with dBASE, Clipper, Alpha IV, FoxBASE+ & FoxPro.
- Portable to DOS, OS/2, UNIX & XENIX.
- Included Windows DLL is linkable with Visual BASIC, Turbo Pascal for Windows, & other DLL-capable compilers.
- Single/multi-user & network access supported.
- 100% C source code (both ANSI & K&R).
- No Royalties. Version 3.0 now shipping.

Complete Source & Windows DLL for only: **\$195** (612) 434-6968



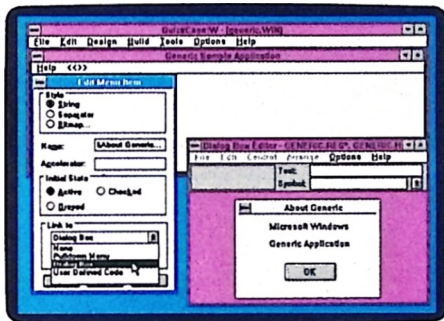
16820 Third Street Northeast
Anoka, MN 55304-4703 U.S.A.
Fax (612) 434-5023

Call about Special
Pricing for Windows DLL
& DOS object code
libraries.

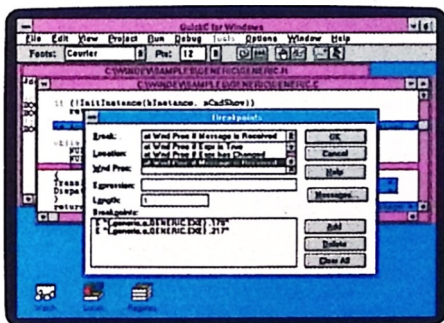
◆ Request 337 on Reader Service Card ◆



Graphic proof that developing Windows apps is now easier.



Use QuickCase:W to draw the graphical elements of your Windows program, from menus and dialog boxes to cursors and icons. Then, let QuickCase:W generate commented expert-level C source code for the design that you've created.



QuickC for Windows supports a wide range of breakpoint types, from breaking at a location to breaking on a Windows procedure when a message is received.

Take a look at Microsoft® QuickC® for Windows™. It's Windows-hosted, so you can edit, compile, and debug inside a single environment. Click on the Toolbar™ to choose frequently-used functions, from changing fonts to setting breakpoints. Workspace templates let you save your screen layouts, so you can reload them quickly from another session. Plus, a remarkable tool known as QuickCase:W lets you create every element in your user interface with a few strokes of a mouse. Then it automatically generates the C source code you need.

All of which lets you create more applications. And more kinds of applications: C programs that call Windows APIs, graphical front-end programs for FORTRAN and COBOL, and C DLLs for other programs.

We suggest a visit to your Microsoft dealer. Because with QuickC for Windows, seeing is believing.

© 1991 Microsoft Corporation. All rights reserved. Printed in the U.S.A. Inside the 50 United States, call (800) 541-1261, Dept. R22; outside the 50 United States, call (206) 936-8661. In Canada, call (416) 568-3503. Microsoft, CodeView, MS-DOS, QuickC and the Microsoft logo are registered trademarks and Windows, and Toolbar are trademarks of Microsoft Corporation. *As used herein, "DOS" refers to MS-DOS and PC-DOS operating systems.

Key Features

- Windows-hosted integrated development environment including an editor, compiler, and debugger.
- All you need to write a Windows-based program.
- QuickCase:W generates source code from your program design and regenerates the code if you change the design.
- Wide range of breakpoint support, including breaking at a location, breaking when an expression has changed, and breaking at a Windows procedure when a message is received.
- Complete printed and online documentation on the Windows API.
- Toolbar for quick access to frequently used editing and debugging functions.
- Save Workspace and Load Workspace for saving and restoring window positions and breakpoints.
- Syntax highlighting in editor.
- Customizable tools menu allows you to run any Windows or DOS* program from within the QuickC for Windows environment.
- QuickWin library for converting well-behaved C programs for DOS to Windows programs.
- Generates Windows Executables, Windows Dynamic Link Libraries, QuickWin Executables for the Windows environment, and MS-DOS* Executables.

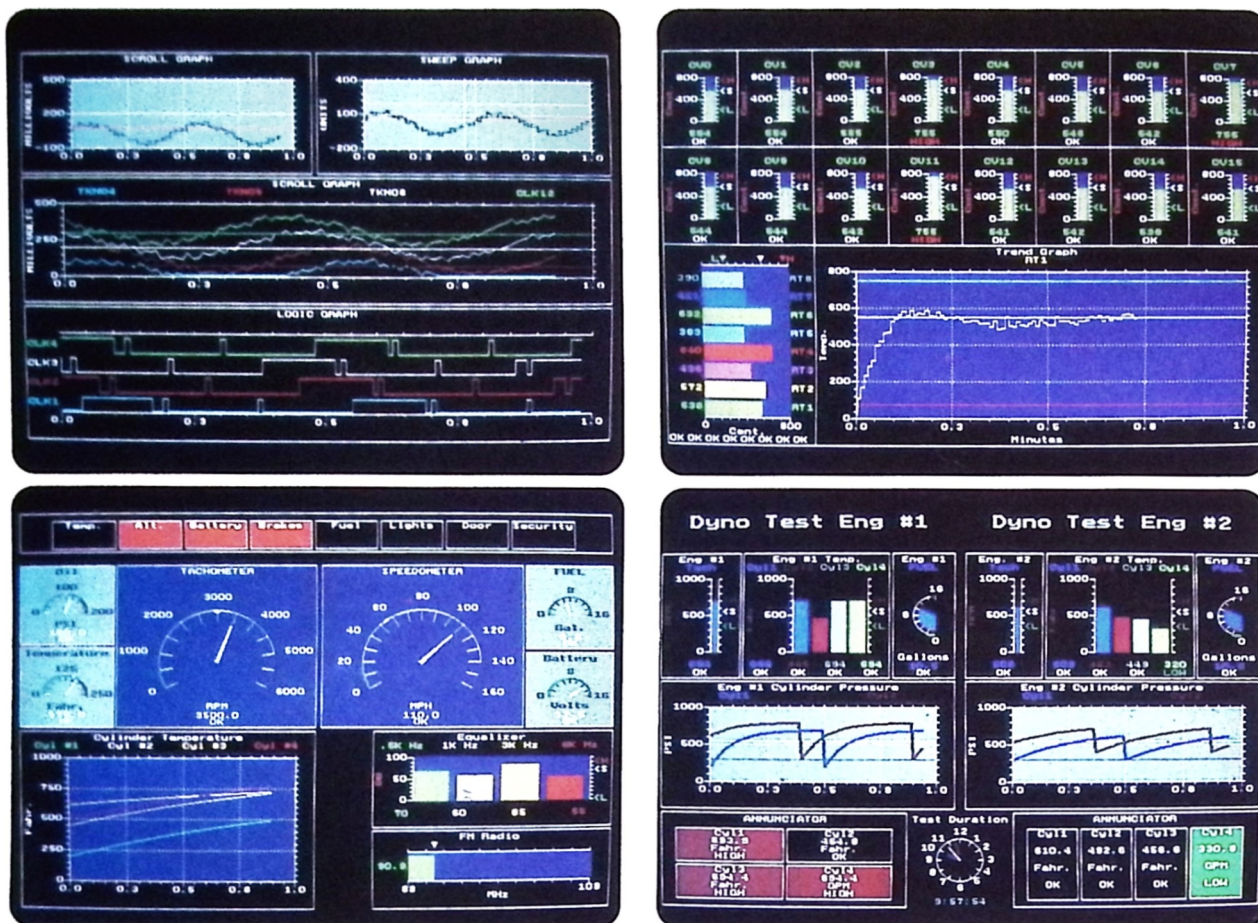
PROGRAMMER'S TIPS

- To rebuild your character-based DOS applications to run under the Windows environment, select the "QuickWin EXE" Project type in the Options menu.
- Use the new and improved Dialog Editor to quickly and easily design dialog boxes for your programs.

Microsoft

Real-Time Graphics and Measurement/Control Tools

Programmers Tools for Microsoft C, Turbo C, C++ and Turbo Pascal



Call or Write for Demo Disk

Fast Real-Time Graphics

Real-time graphics routines for scrolling graphs, sweep graphs, process control bargraphs, annunciator panels, 3 types of meters and general graphics text displays. Real-time mouse routines are also included. The routines are optimized around the graphics primitives which come with the respective compiler.

Powerful Measurement and Control

PID Control (position and velocity algorithms), Thermocouple linearization for the 15 standard TC types (B, BP, BN, E, J, JP, JN, R, K, KP, KN, S, T, TP, TN), Thermocouple curvefitting for any temperature range, Fourier analysis routines including 16K point FFTs.

Source Code and Royalty Free

No additional charge for source code, use these routines in your programs without worrying about royalty fees. No other third party drivers are necessary.

Each version is \$200

Please specify compiler when ordering.

Price includes libraries, source code and a 300 page manual.

Mastercard, Visa accepted. Shipping charge \$5 within USA, \$9 Canada and \$25 elsewhere.

Quinn-Curtis, 21 Highland Circle, Needham, MA 02194 USA Tel. 617/449-6155 FAX 617/449-6109

◆ Request 129 on Reader Service Card ◆

The next step is to replace the CLI interface and compile your code as a Windows application. Unfortunately, even if you prepared ahead for portability there are some problems that may surface.

Types and Structure Alignment

I have already mentioned data types and structure packing. They are especially important issues under Windows. Windows, in its present incarnation, is a 16-bit environment and the default *int* type is 16 bits, but Windows requires structures to be aligned on eight-bit (byte) boundaries. If your code expects structures to be aligned on 16-bit (word) boundaries this may affect you. I ran into alignment problems with a third party database library. The situation forced me into hand padding my structures so members greater than a single byte in size were aligned on word boundaries. I inserted eight-bit padding members of type *char* after an odd number of single byte members.

The Windows programming environment contains many new types defined in the *include* file, *WINDOWS.H*. I recommend you use these types, but confine their usage to the user interface portions of your code.

Memory Models

Since Windows runs under MS-DOS and is subject to the caveats of the Intel segmented architecture, you will have to deal with near and far pointer issues and memory models. Since I wanted as much of my code to be as standard C-like as possible, I tried to avoid sprinkling my code with the keywords *near* and *far* that are not standard C keywords. The general wisdom on Windows claims that programs compiled using the small or medium memory model behave better under Windows than those compiled with the large or compact memory models. Using the small or medium memory model forces you to declare pointers with the *far* keyword if they happen to be in far heap space. Even though using the large and compact memory models is discouraged, many of the Windows library functions also require far pointers as parameters. The only way to get far pointers without adding the *far* keyword to every declaration is to use the large or compact memory model. Windows does not like programs compiled under these memory models because they may contain multiple data segments. Windows fixes multiple data segments in memory. This situation prevents Windows from running more than one instance of such programs and may cause inefficiencies in Windows memory management. This is the case with Microsoft C, but not always with Borland C++. Yes, that is right, the two compilers have a slightly different

implementation of the large and compact memory models. Microsoft C creates multiple data segments when using the large or compact memory model and you have little control over the outcome. Borland C++ creates a single data segment unless you specifically tell it to create more. You can run multiple instances of a program compiled under Borland C++ using the large or compact memory model. The exact program compiled with Microsoft C using the large memory model will run as a single instance only.

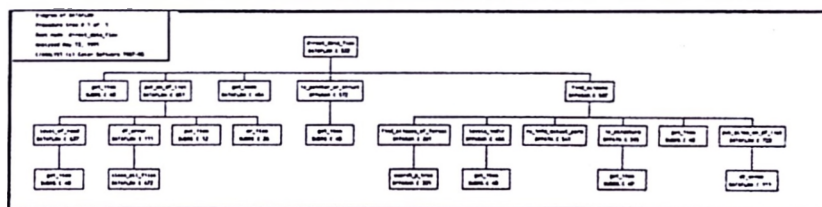
I have tried the large memory model under Windows and did not notice any performance problems with Windows in standard or enhanced mode. I never even tried real mode. Since Windows 3.1 eliminates real mode, I probably never will use real mode. If you need pointers to far data, your choices are to use the large memory model or to use mixed model programming by declaring pointers with the *far* keyword.

Dynamic Memory

Windows does support the *malloc* family of standard C library memory management functions. Unfortunately, they may have slightly different behavior than what you are use to. Depending on the memory model, *malloc* may allocate memory in the near heap. If you want to force allocation in the far heap independent of memory model, you will have to use the function *_fmalloc*. Since Windows maps *_fmalloc* to the Windows function *GlobalAlloc*, there is a limitation on how many times you can call *_fmalloc*. Every time you call

C/ANALYST 2.0

...THE HIGH POWERED DOCUMENTATION TOOL



You have been handed a few inches of source code and have been asked to modify it. C/ANALYST can quickly bring you up to speed. It analyzes your C source and forms a database representing your entire program. From this database, your program can be exhibited from different perspectives.

C/ANALYST is a set of four programs: ANALYZE, XREF, PROCTREE, and DATAFLOW.

ANALYZE creates the database from your source files. A virtual memory manager is used to allow quick access to a large data space. ANALYZE supports ANSI C with extensions for popular compilers. Many customers report that analysis of their source code caught errors they were unaware of.

XREF generates a function index followed by a global cross reference listing. The function index alphabetically lists each function, gives its location and its definition taken from comments in the source code. The global cross reference shows where each object is used and how it is used as well. Read and write codes are included with each reference to a variable, function, or member.

PROCTREE generates tree diagrams which show the overall control flow of the program. Diagrams can be printed sideways on dot matrix printers or the HP Laserjet. Large diagrams can be created for group display or the diagrams can be printed on individual pages with annotated connection circles.

DATAFLOW analyzes and lists possible datapaths between source modules in the program. Including flow through global variables, parameters to functions, return values from functions, and aliases of formal parameters which are pointers.

(503) 581-5622 - CALL TODAY
\$250.00

30 day money back guarantee

CATER SOFTWARE



2182 Westfarthing Way NW
Salem, OR 97304

Add \$15.00 for overseas orders.

◆ Request 491 on Reader Service Card ◆

GlobalAlloc, Windows uses a segment selector. There is a finite number of segment selectors available. This happens to be 8192 for all of Windows – not just your application. If your program requires the allocation of a lot of small pieces of memory, you can quickly run out of selectors even if you still have lots of free memory. The solution to this problem is to call **GlobalAlloc** sparingly and use subsegment allocation. This means you will have to write your own memory manager or buy one of the third party libraries on the market. Version 3.0 of Borland C++ supports subsegment memory allocation and eliminates this problem. I expect eventually all compilers that support Windows will support this feature.

WINSTUB.EXE

Windows allows a non-Windows program to be bound to your Windows program. You specify the stub program in the linker definition file. MS-DOS executes the stub program when you invoke the program from the MS-DOS prompt. I took advantage of this feature to bind the command line or MS-DOS version of a program to the GUI or Windows version of the same program. The only problem I encountered with this was that Borland C++ enforced a 64KB maximum size limitation on the stub program. Microsoft C allowed the stub program to be any size.

UAEs and New Bugs under Windows

When I first compiled my program under Windows as a Windows application, I was extremely disappointed when it would not run without generating UAEs (Unrecoverable Application Errors). Upon tracking down the offending lines of code, a pattern started to emerge. The majority of the UAEs were caused by dereferencing null pointers. This was occurring in the code I had written and also in the standard library code that I was passing null pointers to as parameters. Since the program worked fine under MS-DOS, there was some argument among co-workers about whether these were actual bugs. One of my partners claimed that functions such as *strcmp* should be able to handle a null pointer parameter. Since I could not find any reference that specified how some of the standard library functions responded to null pointers as parameters, I decided to be conservative on this issue and actually modified the program's code to avoid passing null pointers to functions where the behavior was not defined and caused UAEs. I recommend that you be careful about dereferencing null pointers and passing null pointers to standard library functions where the behavior is not specifically defined by the standard or defined in the function description that comes with your compiler's documentation.

Conclusions

Ease in portability between user interfaces requires planning.

A decision to first develop an application with a command line interface and then port it to Windows made me deal head on with GUI portability problems. What I eventually ended up with is a program where most of the code will port to any GUI without rewriting it.

Planning for portability requires you to identify the needed operations and the high level data elements independent of any user interface issues. You should isolate user interface specific code from the core program code. You should be able to access the functionality of your program through a simple command line interface. This can be handy for testing. For a CLI, the *main* function should do nothing but process the command line and make the requested function calls. These same function calls can then be called in a similar way when responding to messages or events in a program with a GUI. The GUI program will have to be more than just a simple *main* function module and may require many new modules that support the GUI functionality and screens. The goal is to have the business end of the code that does the data crunching and calculating remain unchanged when porting from one user interface to another. □



Get **Byte-BOS™** and leave the MULTITASKING to us!

Why spend months designing, documenting, coding, and debugging a multitasking kernel when you can have Byte-BOS, a widely used multitasking operating system on your desk tomorrow?

Byte-BOS Multitasking Operating System consists of a complete set of integrated multitasking components including a robust kernel, "add on" libraries, and development tools that run on a PC.

Only **\$995** buys our multitasking kernel with all these features:

<input type="checkbox"/> preemptive task scheduling	<input type="checkbox"/> resource management
<input type="checkbox"/> task management	<input type="checkbox"/> application code & user manual
<input type="checkbox"/> event management	<input type="checkbox"/> configured to your C compiler
<input type="checkbox"/> message management	<input type="checkbox"/> no royalty C source code
<input type="checkbox"/> timer management	<input type="checkbox"/> 1 year of tech support & updates

Add these multitasking libraries to suit your application:

<input type="checkbox"/> "on chip" serial I/O manager	<input type="checkbox"/> multiple message buffer manager
<input type="checkbox"/> external serial I/O manager	<input type="checkbox"/> multiple timeout manager
<input type="checkbox"/> fixed block memory manager	<input type="checkbox"/> dynamic task manager

Develop and debug Byte-BOS applications on a PC with these tools:

<input type="checkbox"/> BOSPCX for target prototyping	<input type="checkbox"/> BOSVIEW multitasking debugger
--	--

Byte-BOS is available for a wide range of microcontrollers and the PC. Our technical support is responsive, friendly, and knowledgeable. Call us today and see how easy multitasking can be with Byte-BOS.

Byte-BOS™ Integrated Systems
PO Box 3067 Del Mar CA 92014

800-788-7288 or 619-755-8836

◆ Request 136 on Reader Service Card ◆

A Versatile Menu Program for Turbo C

Roger T. Stevens

Introduction

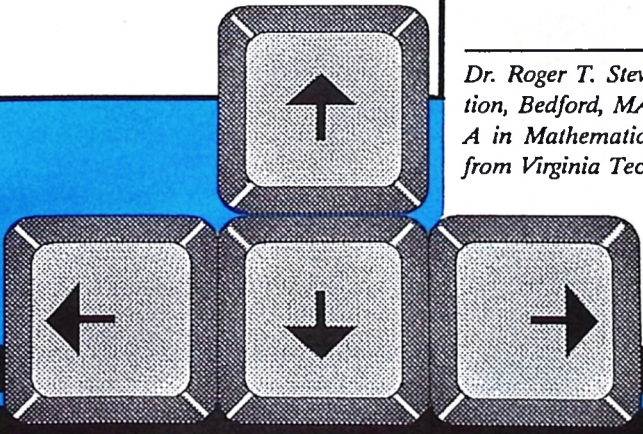
Although there are menu programs available commercially and as shareware, most of them provide only the capability to display and select from a list of menu items, and do not permit interaction with the display. However, interaction with the display is often essential for efficient operation of a program. Look at the sample display of Figure 1. The two bottom lines are provided by the menu function. They give the user the choice of the actions: *DISPLAY DATA*, *CHANGE DATA*, *ADD DATA*, or *QUIT*. All you can do with most menu functions is to select one these actions. However, for the first two of these actions, we want to interact with the display, by selecting a name from the list for the action to be performed upon. The menu function described below will automatically switch mode after an action is selected. For the first two actions, the new mode can permit scanning the list of names with the up and down cursor arrows, highlighting each in turn with a contrasting color combination. The user then hits *ESC* to activate the selected function. The cursor is only permitted to go to the first letter of each name in the list.

Now look at the display of Figure 2. This is a new user generated display which provides detailed data on the person selected from the first display. Again the menu function is activated; this time it just displays two lines of instructions at the

bottom of the screen. This is the display that you see if you chose the action *CHANGE DATA*, from the first display. Those places on the screen where you are allowed to change the data are actually shown on this display in a different color from the rest of the display and the cursor is restricted to only these positions. After you have modified the data in any way you desire, hitting *ESC* returns to the main program. The user can then arrange to read any data changes from the screen into his data files. Note that as long as you are within the menu function, you have the capability to move the cursor to any permissible location and change or rechange the data there. If at the first screen, you selected the *DISPLAY DATA* action, the bottom explanatory lines simply say Hit any key to continue... and no modification of display data is permitted.

For the *ADD DATA* action, no selection from the list of names is permitted; instead the display immediately switches to that of Figure 2, but with the data areas blank, ready to receive data on a new person.

All of the menu and display interaction is controlled by a versatile menu function which provides a number of different modes of operation through passed parameters. This menu function is described in the text that follows, and the code is in Listing 1. The listing also includes a number of useful functions needed for program operation.



Dr. Roger T. Stevens is a member of the technical staff of the MITRE Corporation, Bedford, MA. He holds a B. A. degree in English from Union College, an M A in Mathematics from Boston University, an M. Eng. in Systems Engineering from Virginia Tech, and a PhD. in Electrical Engineering from California Western University. Dr. Stevens' books Graphics Programming in C, Fractal Programming in C, and Fractal Programming in Turbo Pascal are published by M & T Publishing, Inc., 501 Galveston Dr., Redwood City, CA 94063.

Figure 1

List of Names

John S. Jones
Arthur E. Smith
William S. Thompson
Thomas F. Doughty
Edgar Snow
J. Theophilus Johnson
Peter T. Timkins

Select desired menu function with cursor arrows - then hit 'Enter'
DISPLAY DATA CHANGE DATA ADD DATA QUIT

Name List Display

Figure 2

Name: John S. Jones
Address: 132 Main St.
City: Rutland State: VT Zip: 01023
Phone: (802)555-6432

Change data as required:
Then hit 'Esc'.

Display of Detailed Individual Data

What You Can Do with the Menu Program

The menu program begins by displaying two adjacent lines of instructions or menu items on the screen. You can select the location of the first of these lines by setting a parameter called *first_line_loc*, which is passed to the menu function.

The first line normally consists of general instructions; the second line, which is the next line after the first line, can be a list of menu items from which the user makes a selection, or can be another set of general instructions, depending upon the mode of operation of the menu program.

Don't waste any more of your time, let
GENMAKE
take care of all your MAKEFILES !

GENMAKE will:

1. Read all your C modules,
2. Find dependencies,
3. GENerate MAKEfiles.

Features:

Supports most popular MAKE utilities.
Supports any precompiler.
Generates LINK response files.
Supports library modules.
LINK overlay option.
Multiple directories.
DOS and OS/2.

\$139
VISA/MC
30 day MBG

COSYFOR inc.

To order

call: 1-800-267-9367

fax: 514-787-1125

international: 514-485-3434

business hours: M-F 9 to 5 (E.S.T.)

C++ Modeling & Simulation Class Library

Object Oriented Technology for Engineering,
Manufacturing, Finance, Education

Meijin++™ 2.0

- Integration & Approximation ► Optimization & O.R.
- Semi-Persistent Containers ► Queuing Systems
- Discrete Event Simulation ► Exceptions
- Numerical Analysis ► Statistics

More than 110 classes to create models & programs in days.
Documented Source Code, Royalty Free, 700 page Manual
with 80 examples. 30 day Money Back Guarantee.
MS-DOS/WS, Unix V, Sun - Starting: \$349 (Source included)



Network Integrated Services, Inc.

221 West Dyer Road
Santa Ana CA 92707-3426
Fax: (714) 433-2347

CALL

(714) 755-0995



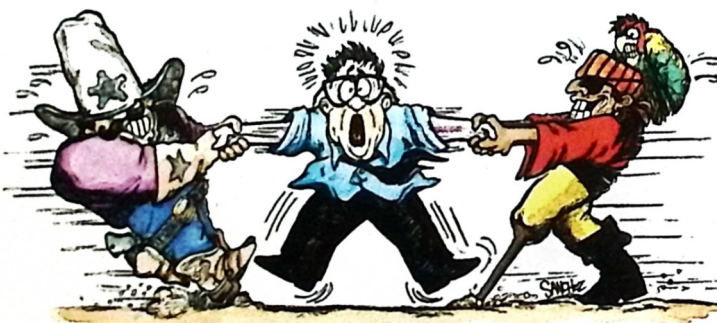
The best C/C++
tools for Windows are
from the company that
makes Windows.

Table 1

BACKGROUND	Black	Blue	Green	Cyan	Red	Magenta	Brown	Light Gray
FOREGROUND								
Bright White	15	31	47	63	79	95	111	127
Yellow	14	30	46	62	78	94	110	126
Light Magenta	13	29	45	61	77	93	109	125
Light Red	12	28	44	60	76	92	108	124
Light Cyan	11	27	43	59	75	91	107	123
Light Green	10	26	42	58	74	90	106	122
Light Blue	9	25	41	57	73	89	105	121
Dark Gray	8	24	40	56	72	88	104	120
Light Gray	7	23	39	55	71	87	103	119
Brown	6	22	38	54	70	86	102	118
Magenta	5	21	37	53	69	85	101	117
Red	4	20	36	52	68	84	100	116
Cyan	3	19	35	51	67	83	99	115
Green	2	18	34	50	66	82	98	114
Blue	1	17	33	49	65	81	97	113
Black	0	16	32	48	64	80	96	112

Color Table

TRAPPED BETWEEN PIRACY AND PROTECTION?



Vault has it all. Why go anywhere else? Use our patented disk-based products and make yourself a hero by adding as much as 60% of your copy protection dollar to your bottom line. Your customers will love you for it - no more misbehaving printers, sleeping output ports, clunky dongles on laptops.

There is an inexpensive, no hassle solution. If you object to copy protection because of its history of incompatibility and you don't want to make your lawyer rich suing your biggest customer, try our dongle on a disk combo. It automatically checks the hard disk for an installed program, the parallel port for a Romlok, and finally the disk drive for a Prolok disk. And, it works every time. We guarantee it.

- Prolok Patented disk-based copy protection
- Romlok Hardware security device
- Nolok Prevents disassembly or modification of your code
- Unilok Protection on networks and frequently updated software.
- Chronolok Execution counter for perishable or demonstration software
- Netlok The generic network management and control system

*Easy to use *No source code changes required. *100% compatible. *Patented technology. *Supports duplication equipment. *Linkable object modules for custom security. *Hard disk installable. *Solutions for every operating system environment. *Fully functional evaluation kits available OS, and Windows support.

PROFIT PROTECTION *Prolok*

P.O. Box 820, Agoura, CA 91376



(800) 366-8285 (818) 889-6719 FAX (818) 889-1465

◆ Request 162 on Reader Service Card ◆

Introducing Microsoft C/C++.

	Microsoft	Borland*
Windows Class Libraries	C/C++ 7.0	BC++ 3.0
Covers entire Windows API	Y	N
Menu support	Y	N
GDI support	Y	N
OLE 1.0 support	Y	N
Exception handling	Y	N
Diagnostics support	Y	N

Code Generation: DES Encryption Test	C/C++ 7.0	BC++ 3.0
EXE size	5K	73K
Execution time	820 sec	1500 sec

BYTE Build Test	C/C++ 7.0	BC++ 3.0
Using fast compile, pre-compiled headers	300 sec	420 sec
Optimized EXE size	162.4K	202.6K

Compiler Features	C/C++ 7.0	BC++ 3.0
Code in pre-compiled headers	Y	N
Inline any C/C++ code	Y	N
Auto-inlining	Y	N
P-code	Y	N

Windows Tools	C/C++ 7.0	BC++ 3.0
Windows resource editing tools	Y	Y
Profiler for Windows & MS-DOS	Y	Y
Windows Help compiler	Y	Y
Windows setup builder	Y	N
Total documentation	5408 pp	4038 pp
Windows 3.1 debug kernel	Y	\$199 extra
Total Price*	\$495	\$948+

By almost any measure, new Microsoft C/C++ Version 7.0 development system for Windows™ is the best way to create all your applications for the Windows and MS-DOS® operating systems.

With better code generation and pre-compiled headers, you'll have all the tools you need to write better code, faster.

And because the Microsoft Foundation Classes have the most complete framework for Windows, you'll use the same building blocks for your products that we use for ours.

C/C++ 7.0 also includes the Windows 3.1 debugging kernel which can help you find the bugs. Plus, all the tools you'll ever need to edit your resources, compile the help files and even build your very own graphical setup programs for Windows.

Judge for yourself. Try new Microsoft C/C++ 7.0 and, as a Microsoft, Borland or Zortech customer, you'll be able to upgrade for just \$139*— and for a limited time, you'll get a free copy of Qualitas® 386MAX™ in the box!

So call your local reseller now, or call Microsoft at (800) 541-1261, Department Z69. Get your tools from the people who make Windows, because we've been building Windows tools longer.



Upgrade for just \$139!

Microsoft®

*Reseller prices may vary. Offer good only in the 50 United States. ©1992 Microsoft Corporation. All rights reserved. Printed in the USA. In the 50 United States, call (800) 541-1261, Dept Z69. For information only: In Canada, call (800) 563-9048; outside the 50 United States and Canada, call (206) 936-8661. Microsoft and MS-DOS are registered trademarks and Windows is a trademark of Microsoft Corporation. All comparisons are with Borland's C++ Compiler and Application Frameworks version 3.0. Benchmarks run by third parties; details available on request. Borland is a registered trademark of Borland International, Inc. Qualitas is a registered trademark and 386MAX is a trademark of Qualitas, Inc.

Listing 1

```

/*
menu = controls cursor movement and menu display and selection -
returns the number of the selected menu item.

values:          a string consisting of two digits showing
                  the number of menu choices, followed by
                  four digits for each choice, the first
                  two showing the starting column of the
                  menu item and the second two its length.

menu_first_line: the first line of instructions in the
                  menu mode.

menu_second_line: the second line in the menu mode. It con-
                  tains the menu choices.

screen_first_line: the first line of instructions in the
                   screen mode.

screen_second_line: the second line of instructions in the
                   screen mode.

menu_type:        0: Start with Menu Mode.
                   Alphanumeric characters are ignored
                   when entered in screen mode.
                   1: Start with Menu Mode.
                   Alphanumeric characters are displayed
                   when entered in screen mode
                   2: Start with Screen Mode.
                   Alphanumeric characters are ignored
                   when entered in screen mode.
                   3: Start with Screen Mode.
                   Alphanumeric characters are displayed
                   when entered in screen mode.

screen_color:     color for screen mode.
menu_color:       color for menu display.

highlight_color:  color of selected menu item

select_no:        number of menu items using selection

escape_char:      number of the key selected for escape
                  from the screen display.

map:              bit map of permitted cursor locations.
                  The cursor will go to permitted locations
                  only and no others.
*/

int menu(char values[],char menu_first_line[],char menu_second_line[],
char screen_first_line[],char screen_second_line[],int menu_type,
int screen_color,int menu_color,int highlight_color, int select_no,
int escape_char, int first_line_loc, char map[25][10])
{
    union REGS reg;
    int i,choices,indx,start,length,menu_second_line_length;
    int interim,remainder,temp;
    char spaces[80],prev_char;

    for (i=0; i<76; i++)
        spaces[i] = ' ';
    spaces[76] = '\0';
    menu_second_line_length = strlen(menu_second_line);
    gotoxy(2,first_line_loc);
    choice = 1;
    if (menu_type <= 1)
    {
        color_printf("%s",menu_color,menu_first_line);
        gotoxy(2,first_line_loc+1);
        length = values[4] - '0';
        length = 10 * length + values[5] - '0';
    }
}

```

You can specify the color combination for the menu items and the color combination which is used to highlight the currently selected menu item. The rest of the screen display remains as you generated it before calling the menu function. A menu item is selected by use of the cursor arrows; when selection is complete, this phase of the menu program is terminated by hitting the *Enter* key. For any number of menu items, beginning with the leftmost, you can specify an alternate (screen) mode of operation, which is entered after the *Enter* key is hit. You can specify each permissible cursor position for this alternate mode, and the cursor will only be allowed to go to these selected positions. (For example, if the down arrow is hit, the menu function will look at the current column and the next line and move the cursor there if it is a permissible position. If that position is not permissible, the function will look for the nearest permissible position on that line; if there is none, it will look at the next line, and so forth until a permissible location is found.) You can also specify two lines of text which will appear on the menu lines when the alternate mode is entered. You can specify whether this alternate mode of operation will be a *select* or an *enter text* type of operation. If selection is chosen, the text from the cursor to the next occurrence of two adjacent spaces is highlighted. Usually for this type of operation, the only allowable cursor positions are at the beginning of each selectable item on the display, so the entire selected item is highlighted.

If you specified the *enter text* type of operation, the user may enter alphanumeric data in any permissible cursor location. You may specify the color combination which this entered data will have. When data entry is complete, the same escape character specified above is used to terminate the screen mode of operation.

Determining Permissible Cursor Positions

The heart of the menu program is the capability to specify which positions on the screen the cursor is permitted to occupy. When in the selection mode, there should only be one permissible cursor position for each item to be selected. That is usually the first character of the

item description. When changing or entering data on the screen, a file structure is usually determined, which specifies the names of the data items for each file entry and the length of each of these items. A display location and length is established for the display of each item, and the cursor is only allowed to go to the allocated space for each item. These areas of the screen can then be read to obtain the modified data. By prohibiting the cursor from going to other areas, it becomes impossible for the user to overwrite item definitions or to enter data that is too long to fit into the file.

The permissible cursor positions are controlled in the menu program by use of a bit map, which contains one bit for each character position on the screen. The bit map consists of a character array of 25 by 10 characters. The 25 is for the 25 screen lines; the 10 is for ten bytes of eight bits each to provide for the 80 character positions on a line.

If a particular bit is one, the cursor is permitted to go to that location; if it is a zero, the cursor is prohibited from going there. The bit map for a particular menu

Listing 1 — Cont'd

```
for (indx = 0; indx < menu_second_line_length; indx++)
    if (indx < length)
        putcolorchar(menu_second_line[indx],
            highlight_color);
    else
        putcolorchar(menu_second_line[indx],
            menu_color);
}
else
{
    color_printf(screen_first_line,menu_color);
    gotoxy(2,first_line_loc+1);
    color_printf(screen_second_line,menu_color);
}
choices = 10 * (values[0] - '0') + values[1] - '0';
gotoxy(column,row);
for(;;)
{
    key_id = getch();
    if (key_id == 0)
        key_id = getch()+256;
    if (menu_type <= 1)
    {
        switch(key_id)
        {
            case 13:
                if(choice > select_no)
                    goto ExitPoint;
                change_line_color(47);
                gotoxy(2,23);
                color_printf("%s",screen_color,spaces);
                gotoxy(2,23);
        }
    }
}
```

4
Configurations
Available

The Heart of the Matter...

RTXC™

Real-Time Multitasking Executive

- INTEL 80x88/x86, 80x96, 80x51 • HITACHI 6303
- MOTOROLA 680x0, 683xx, 68HC11, 68HC16
- INMOS T400, T800 • ZILOG Z80/Z180

- Preemptive Scheduling
- Fixed or Dynamic Priorities
- Timeout on some services
- Configurable and ROMable
- Intertask Communications
- Messages
- Queues
- Semaphores
- Memory Management
- Resource Manager
- Over 50 Executive Services Available
- System Level Debugging Utility
- System Generation Utility
- Written in C
- Source Code Included
- No Royalties
- Technical Support
- Broad C Compiler Support
- Sensible License Agreement
- 450+ Page User's Manual

Ask about ASSIST™
Our new PC Development
Package Combination

One Time License Fee From \$995
Discounts for Multiple Licenses/Ports
The only real-time kernel you'll ever need™

A.T. BARRETT & ASSOCIATES, INC.
11501 Chimney Rock, Houston, TX 77035
FAX 713/728-1049
Phone 800/525-4302 or 713/728-9688

FREE DEMO
AVAILABLE

◆ Request 380 on Reader Service Card ◆

QUINN COMMUNICATIONS
509 VOSBURG ROAD • WEBSTER NEW YORK 14580

C++ IMAGING

```
main()
{
    //lena is 512x512x256
    vimage a["lena.tif"];
    a.autoscale();
    a.sharpen();
    vimage b = a.copy(0,0,255,512);
    //produce "negative" left side
    b.remap_linear(-1,255);
    a.paste(0,0,b);
    a.vga_scroll();
    a.save("remap.pcx");
}

main()
{
    vimage a["lena.tif"];
    //remove face from image
    vimage b = a.cut(256,200,384,328);
    //magnify by interpolation
    vimage c = b.interpolate(512,512);
    //write out to HP print file
    c.hp_screen(200,200,300,"out.hp");
}
```



HP Printing + Convolution + Remap + Resize + Cut +
Copy + Paste + Color + Super VGA + Operator Overloads
+ PCX/TIFF + Rotate + More! (Over 70 functions)

CALL NOW FOR FREE BROCHURE!

SOURCE \$349 • LIBRARY \$229 • 30 DAY GUARANTEE • VISA • MASTERCARD
For BORLAND C++

ImagingObjects

1-800-253-7804 or 1-716-671-7998

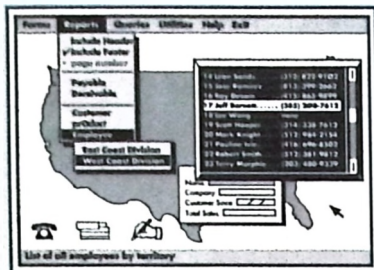
◆ Request 154 on Reader Service Card ◆

IF YOU PROGRAM

in C, C++, Basic, Fortran, Cobol,
Pascal, dBase, **

HI-SCREEN Pro II™

is your user interface solution



With HI-SCREEN Pro II, you can design modern text and graphic user interfaces in a snap:

- Design interface objects using interactive editors. Generate all windows, icons, menus, and data entry screens in a WYSIWYG fashion.
- Test all objects directly under the editors, including data validation, menus, icons and dialog boxes.
- Manage objects from your program using a set of versatile and powerful functions.

Hercules, CGA, EGA, VGA, 25/30/43/50 line modes (auto-detect) • Full mouse support & keyboard control • Automatic data validation • Screen editor functions include: view, load, merge, copy, delete, color, draw (using any character), frame (standard, 3D, round corners), undo, on-line ASCII table, merge graphic screens & icons for hypertext effects, direct access to other editors (icon editor, menu editor, graph editor, etc.) • Field types: alphanumeric, real, integer, long text, chosen characters, date, time, list, mouse box, selector, password, flag button • Field description: type, range, format, access key, justification, color, help message or screen, etc. • Data entry test mode under the editor • Automatic context-sensitive help systems • Automatic scrolling windows with scroll bars & boxes • Automatic shadowing effects • Combine text and graphics • Capture any text and graphic screens • Flexible input management: field-by-field or full-screen, interrupt input before or after field, run parallel task, check current field, etc. • Automatic menu systems: pull-down, horizontal, vertical • Link screens into your .EXE or keep a separate library file • Only \$395 • No royalties • Call for OS/2 and Windows 3 modules.

CALL: 1-800-338-2852



See us at
Booth
#909

Softway, Inc.

185 Berry Street, Suite 5411, San Francisco, CA 94107

Tel.: 415-896-0708 • Fax: 415-896-0709

Listing 1 — Cont'd

```

color_printf("%s", menu_color,
screen_first_line);
gotoxy(2,24);
color_printf("%s", screen_color, spaces);
gotoxy(2,24);
color_printf("%s", menu_color,
screen_second_line);
gotoxy(column,row);
menu_type += 2;
break;
case 333: /*Right Arrow*/
choice = choice + 2;
case 331: /*Left Arrow*/
--choice;
if (choice < 1)
choice = choices;
if (choice > choices)
choice = 1;
start = 10 * (values[(choice-1)*4+2] - '0')
+ values[(choice-1)*4+3] - '0';
length = 10 * (values[(choice-1)*4+4] - '0')
+ values[(choice-1)*4+5] - '0';
gotoxy(2,24);
for (indx = 0; indx < menu_second_line_length;
indx++)
{
if ((indx >= start) && (indx < start
+ length))
putcolorchar
(menu_second_line
[indx],
highlight_color);
else
putcolorchar
(menu_second_line
[indx],
menu_color);
}
gotoxy(column,row);
break;
default:
if ((key_id >= 0x41) && (key_id <= 0x7A))
{
temp = toupper(key_id);
for (indx=0; indx<choices; indx++)
{
start = 10 * (values[
(indx-1)*4+2] -
'0')+values[(indx
-1)*4+3] - '0';
if (temp == menu_second_line
[start])
{
choice = indx;
if(choice >
select_no)
goto ExitPoint;
change_line_color
(47);
gotoxy(2,23);
color_printf("%s",
screen_color,spaces);
gotoxy(2,23);
color_printf("%s",
menu_color,
screen_first_line);
gotoxy(2,24);
color_printf("%s",
screen_color,spaces);
gotoxy(2,24);
color_printf("%s",
menu_color,
screen_second_line);
gotoxy(column,row);
menu_type += 2;
}
}
}
}

```


may be generated dynamically or it may be generated manually by determining what bits need to be set and where they are located in the array. Since this latter process can be rather cumbersome, a utility function, *set_cursor*, has been provided to do the job automatically. You temporarily insert *set_cursor* into your program, just after the display has been generated. You can now move the cursor around the display and insert an *x* or *X* at every position where the cursor is to be allowed. You should also replace any *x* or *Xs* that occur naturally in the display in locations prohibited to the cursor with some other character. Make sure not to hit the *Enter* key until you have inserted all of the required *xs* in the display.

When you hit *Enter* the program reads the entire screen and generates a file called *MATRIX.C*, which contains all of the ASCII data needed for initializing a cursor map array in your program. You can set up the bit map array by inserting the following line in your program:

```
char nnnnnnnn[25][10] =
```

where nnnnnnnn is the name of your bit map. If you are using the Turbo C total environment editor, place the cursor after the equals sign and type *^KR*. When asked for the file name, type in *MATRIX.C*. The required data for the bit map will then be inserted into your program. You can then remove *set_cursor* from your listing and recompile the program. When you run the menu function, you will find that the cursor will only go to those positions which you marked with an *x* or *X* when you used *set_cursor* to construct the bit map.

Key Designations

Normal keys on the IBM PC keyboard generate the standard ASCII representation of the selected letter or number. Most special keys, such as the cursor arrow keys and the F1 through F10 function keys return two characters, first a hex 0 and then some number from 1 to 255. To put all key returns into a common format, the menu program automatically reads a second character from the keyboard when the first character is 0 and adds 256 to this second character to obtain a unique

Listing 1 — Cont'd

```

    }
}
else
{
    if (key_id == escape_char)
        goto ExitPoint;
    switch(key_id)
    {
        case 8: /*Backspace*/
        case 33: /*Left Arrow*/
            if ((menu_type == 0) ||
                (menu_type == 2))
                change_line_color
                (screen_color);

            do
            {
                column--;
                if (column < 0)
                {
                    column = 79;
                    row --;
                    if (row < 0)
                        row = 24;
                }
                indx = column/8;
                remainder = column - indx*8;
                interim = map[row][indx] &
                    (0x01 << remainder);
            }
            while (interim == 0x00);
            gotoxy(column,row);
            if ((menu_type == 0) ||
                (menu_type == 2))
                change_line_color
                (highlight_color);
    }
}

```

"Are you taking advantage of the tremendous power and flexibility of C++?"

M++ CLASS LIBRARY

Unleash the Power of C++

The powerful array handling capabilities that have accelerated development in advanced scientific languages for years are now available in C++. The M++ library is a complete, standardized, object-oriented multidimensional array language extension to C++. With M++ your memory and array handling problems are gone. Instead you have a powerful environment for developing tight, fast applications using advanced numeric methods. Use the M++ classes directly, or as a solid base for your C++ class development.

M++ incorporates the linear systems and eigensystem techniques of LINPACK and EISPACK. Optional M++ modules further extend C++ into advanced object-oriented statistical scientific programming environments. Modules available are:

SUM - Statistical Utilities Module

LSM - Least Squares Module

OPTIM - Optimization Module

TEST - M++ Test Suite

QUAD - Numeric Integration Module

M++ is Portable

M++ is available on platforms that support AT&T 2.0 and 2.1 compatible compilers. Call for information on DOS, UNIX and other platforms.

dyad
SOFTWARE

515 116th Avenue NE
Suite 120
Bellevue, WA 98004
(800) 366-1573
(206) 637-9428 FAX

Listing 1 — Cont'd

```
        break;
    case 333: /*Right Arrow*/
        if ((menu_type == 0) ||
            (menu_type == 2))
            change_line_color
            (screen_color);
        do
        {
            column++;
            if (column > 79)
            {
                column = 0;
                row ++;
                if (row > 24)
                    row = 0;
            }
            indx = column/8;
            remainder = column - indx*8;
            interim = map[row][indx] &
                (0x01 << remainder);
        }
        while (interim == 0x00);
        gotoxy(column,row);
        if ((menu_type == 0) ||
            (menu_type == 2))
            change_line_color
            (highlight_color);
        break;
    case 13:
        column = 0;
    case 336: /*Down Arrow*/
        if ((menu_type == 0) ||
            (menu_type == 2))
            change_line_color
```

number that will not duplicate one of the normal ASCII codes. This key input is stored in a variable called *key_id*. Thus, when looking at the program listing, some of the comparisons of *key_id* with various numbers may appear unfamiliar. They can be identified by taking any chart of keyboard codes and adding 256 to the second character generated by a particular key. The menu program has the flexibility of specifying which key will be used to escape from the screen type of operation. This capability will be described in further detail below. You should note, however, that whatever character you select for escape cannot be entered as data on the screen.

Menu Options

The programmer has almost unlimited flexibility in defining how the menu program is to be used. The menu options are selected by parameters passed through the menu function.

The first parameter passed to the menu function is a string called *values* which defines the characteristics of the menu line. It begins with two digits which define the number of menu function items. Four digits then follow for each menu item. The first two represent the column of the display at which that menu item begins. The next two represent the number of characters in the menu item. These four digit entries must correspond to the actual spacing of the entries in the menu line string described below.

The next parameter, called *menu_first_line*, is a string showing the first line of instructions when the program is showing the menu type display. Following that is a string called *menu_second_line*, which is the second line of instructions for the menu type (the actual set of menu function items). Next are two strings called *screen_first_line* and *screen_second_line*, which are the first and second lines of instructions when the menu program switches to the second type of operation.

The next parameter, called *menu_type*, selects the mode of operation. There are four modes of operation for the menu function, which are

PERENNIAL™ VALIDATION SUITES

- ANSI/FIPS C Validation
- C++ Validation
- UNIX SYSTEM V Validation
- BSD 4.3 Validation
- POSIX/FIPS 151-1
Conformance Testing

408/748-2900



PERENNIAL

4699 Old Ironsides Drive, Suite 210
Santa Clara, California 95054

◆ Request 108 on Reader Service Card ◆

Listing 1 — Cont'd

```

        (screen_color);
    row++;
    if (row > 24)
        row = 0;
    indx = column/8;
    remainder = column - indx * 8;
    interim = map[row][indx] & (0x01 <<
    remainder);
    if (interim != 0x00)
    {
        gotoxy(column,row);
        if ((menu_type == 0) ||
            (menu_type == 2))
            change_line_color
            (highlight_color);
        break;
    }
    column = 0;
    do
    {
        indx = column/8;
        remainder = column - indx*8;
        interim = map[row][indx] &
        (0x01 << remainder);
        if (interim != 0x00)
        {
            gotoxy(column,row);
            if ((menu_type == 0)
                || (menu_type == 2))
                change_line_color
                (highlight_color);
            break;
        }
        column++;
        if (column > 79)
        {
            column = 0;
            row ++;
            if (row > 24)
                row = 0;
        }
    }
    while (interim == 0x00);
    break;
case 328: /*Up Arrow*/
    if ((menu_type == 0) ||
        (menu_type == 2))
        change_line_color
        (screen_color);
    row--;
    if (row < 0)
        row = 24;
    indx = column/8;
    remainder = column - indx * 8;
    interim = map[row][indx] & (0x01
    << remainder);
    if (interim != 0x00)
    {
        gotoxy(column,row);
        if ((menu_type == 0) ||
            (menu_type == 2))
            change_line_color
            (highlight_color);
        break;
    }
    column = 79;
    do
    {
        indx = column/8;
        remainder = column - indx*8;
        interim = map[row][indx] &
        (0x01 << remainder);
        if (interim != 0x00)
        {
            gotoxy(column,row);
            if ((menu_type == 0)

```

Do you use C++?

Running out of memory?

Need some industrial- strength, case-hardened, Swiss-army- knife classes?

Oh.

Have you heard about VMO?

Could you use a doubly-linked
list class that'll handle monster
objects a gigabyte long?
(Is there a BLOB in the house?)

What about *really* big arrays?
(Like jaw-dropping, eye-pop-
ping, need-new-shorts big?)

Or virtual, dynamic allocation?
(Ooh, *look*, Toto! 0 to 4gb and
not a single malloc or free!
We *can't* be in C anymore!)

Any questions?

Zircel Software (303) 750-9543

Compuserve box 76326, 3072
✦ Request 433 on Reader Service Card ✦



REAL-TIME MULTITASKING KERNEL

8086/88, 80x86/88 80386
Z80, 64180, 8080/85 68000/10/20

- Fast, reliable operation
- Compact and ROMable
- PC peripheral support
- DOS file access
- C language support
- Preemptive scheduler
- Time slicing available
- Configuration Builder
- Complete documentation
- Intertask messages
- Message exchanges
- Dynamic operations
 - task create/delete
 - task priorities
 - memory allocation
- Event Manager
- Semaphore Manager
- List Manager
- InSight™ Debugging Tool

THE BEST

Join over 600 developers such as
IBM®, Xerox, Hewlett Packard,
Hayes, Hughes Aircraft and NASA.

CHOOSE AMX

The best low-cost, high-performance
real-time multitasking system
available today.

No Royalties
Source Code Included

Demo Disk and
Manual only **\$75 US** Call for prices for
AMX 86 **\$3000 US** other processors.
(Shipping/handling extra)

IBM is a registered trademark of IBM Corp.
Z80 is a trademark of Zilog, Inc.
AMX, AMX 86, InSight are trademarks of
KADAK Products Ltd.

KADAK Products Ltd.

206-1847 West Broadway
Vancouver, B.C., Canada
V6J 1Y5



Telephone: (604) 734-2796
Fax: (604) 734-8114

Listing 1 — Cont'd

```

        || (menu_type == 2))
        change_line_color
        (highlight_color);
        break;
    }
    column--;
    if (column < 0)
    {
        column = 79;
        row --;
        if (row < 0)
            row = 24;
    }
    while (interim == 0x00);
    break;
default:
    if ((menu_type == 1) ||
        (menu_type == 3))
    {
        putcolorchar(key_id,
            screen_color);
        do
        {
            column++;
            if (column > 79)
            {
                column = 0;
                row ++;
                if (row > 24)
                    row =
                        0;
            }
            indx = column/8;
            remainder = column -
                indx * 8;
            interim = map[row]
                [indx] & (0x01
                    << remainder);
        }
        while (interim == 0x00);
        gotoxy(column,row);
    }
}

ExitPoint:
return choice;
}

/*
change_line_color = changes the color of a line up to a double space
*/

void change_line_color(int color)
{
    union REGS rin;
    char prev_char;

    prev_char = 's';
    for(;;)
    {
        rin.h.ah = 3;
        rin.h.bh = 0;
        int86(0x10,&rin,&rin);
        ch = read_char_from_screen();
        if((ch == ' ') && (prev_char == ' '))
            break;
        prev_char = ch;
        gotoxy(rin.h.dl,rin.h.dh);
        putcolorchar(ch,color);
    }
    gotoxy(column,row);
}

/*
clearscreen = clears the screen and displays selected color background
*/

```


controlled by this parameter. They are assigned by one of the numbers zero to three. The modes of operation are:

0 = When the menu program is called, it will start with the menu display. When it is in the screen display type of operation, all alphanumerics will be ignored.

1 = When the menu program is called, it will start with the menu display. When it is in the screen display type of operation, all alphanumerics will be displayed where typed.

2 = When the menu program is called, it will start with the screen display. When it is in the screen display type of operation, all alphanumerics will be ignored.

3 = When the menu program is called, it will start with the screen display. When it is in the screen display type of operation, all alphanumerics will be displayed where typed.

The next parameter, *screen_color*, is the screen color. This has no effect upon the display that has already been generated, but does determine what color combination will be used for alphanumerics typed on the screen and to restore the background for that part of the two instruction lines that is not used. Normally it should be the same color combination used in generating the original display screen. Table 1 shows the color combinations represented by each number. The next parameter, called *menu_color*, is the color combination used by the two lines of instructions produced by the menu. The next parameter, called *highlight_color*, is the color combination used to highlight the selected menu item.

The next parameter determines how many menu functions will switch to the screen type of operation when selected. Those functions which switch to the screen type of operation must be grouped at the beginning of the menu line, since the count in this variable begins with the leftmost function.

The next parameter, called *escape_char*, is the representation of the key which must be hit to escape from the screen type of operation. It is a number which is the ASCII value produced by a regular key or the value plus 256 if the keyboard output is a two character output beginning with 0. Thus any keyboard output may be selected.

Listing 1 — Cont'd

```
*/
void clearsreen(int color)
{
    int indx;
    union REGS reg;

    gotoxy(0,0);
    reg.h.ah = 9;
    reg.h.al = 0x20;
    reg.h.bh = 0;
    reg.h.bl = color;
    reg.x.cx = 2000;
    int86(0x10,&reg,&reg);
}

/*
color_printf = printf with selected foreground and background colors
*/

void color_printf (char *msg,int color,...)
{
    union REGS reg;
    char ch, string[2000];
    int i = 0;
    va_list (ap);

    va_start (ap,msg);
    vsprintf(string,msg,ap); /*do printf to string*/
    va_end(ap);
    while ((ch=string[i++]) != '\0') /*get chars from string till end*/
    {
        if (ch == 0x0A) /*is character a line feed*/
        {
            reg.h.ah = 3;

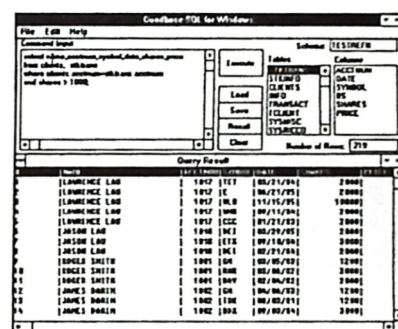
```

Quadbase-SQL/Win™
allows you to build industrial strength database applications under Windows using standard SQL with your favorite choice of Windows front-end development tools such as Visual Basic, SQLWindows, Smalltalk/V Windows, ACTOR, Toolbook, Windows Maker, Microsoft C etc.. The system offers 1) a very fast, compact and reliable SQL engine as a DLL, 2) an interactive query tool written in Visual Basic and 3) a report writer.

Call now and find out why **GE, Compaq Computer, Microsoft, Price Waterhouse, Reuters** and many top notch companies are our customers.

dQUERY 4.0

is your power tool for ad hoc querying, report writing, and building canned query systems using SQL and QBE.



- Fully supports ANSI SQL-86 level 2 standard, outer-join, primary key and referential integrity, scroll cursor, transaction processing and security features, and record locking for multi-user access
- Especially designed to handle large files
- Embedded SQL preprocessor for C
- Report-writer
- Supports dBASE, 1-2-3, Foxbase, and Clipper file and index formats
- dQUERY™ 4.0 is included in Quadbase-SQL/Win

Call for a free demo disk.



Quadbase Systems Inc.
790 Lucerne Drive #51
Sunnyvale, CA 94086
Voice: (408) 738-6989
Fax: (408) 738-6980

* Other trademarks appearing in this ad are trademarks of their respective companies.

Fortify C Programs With A Dynamite User Interface

Vitamin C advances the development process with everything you need to build a state-of-the-art user interface into every program you write.

A robust library of C functions helps you fly through complex applications and come out looking great. One simple function call opens a window. Dozens of other high level essentials are just as easy. A complete set of low level routines gives you total control to fine tune the details. No other user interface library is as intuitive or easy to use.

- ♦ *Windows*
- ♦ *Menus*
- ♦ *Entry fields*
- ♦ *Push buttons*
- ♦ *Radio buttons*
- ♦ *Check boxes*
- ♦ *List boxes*
- ♦ *Help*
- ♦ *Mouse*

Vitamin C's open-ended design provides virtually limitless customization potential without modifying the source. Special filters let you intercept and modify operations as needed. The ultimate in programming flexibility.

Versions for DOS, OS/2, Unix, Xenix and VMS start at \$395 including free source code and phone support. No royalties or runtime fees. Just clean, fast code with full documentation.

*Free
demos & info
via our
24-hour BBS!*

*(214)
245-9518*

So look like a genius. Vitamin C gives you freedom to be creative and power to get more from your programming efforts. Vitamin C—everything you need to fortify your programs with a dynamite user interface. Try it today!

Order Hotline: (800) 726-6447

Creative Programming

Box 112097
Carrollton, TX 75011 USA
(214)245-9139 Fax (214)245-9717



Listing 1 — Cont'd

```
int86(0x10,&reg,&reg);
/*get cursor position*/
reg.h.dl = 0;
reg.h.dh++;
/*cursor value to beginning of next line*/
reg.h.ah = 2;
int86(0x10,&reg,&reg);
/*set new cursor position*/

}
else
{
    reg.h.ah = 9;
    reg.h.al = ch;
    reg.x.bx = color;
    reg.x.cx = 1;
    int86(0x10,&reg,&reg);
    /*send a color character to display*/
    reg.h.ah = 3;
    int86(0x10,&reg,&reg);
    /*get cursor position in D reg*/
    reg.x.dx++;
    reg.h.ah = 2;
    /*increment cursor position value*/
    int86(0x10,&reg,&reg);
    /*set cursor to new position*/
}
}

/*
gotoxy = moves cursor to selected column and row
*/

void gotoxy(int col, int row)
{
    union REGS reg;
    reg.h.ah = 2;
    reg.h.bh = 0;
    reg.x.dx = (row << 8) + col;
    int86(0x10,&reg, &reg);
}

/*
putcolorchar = displays a character with selected color foreground
and background
*/

void putcolorchar(char character, int color)
{
    union REGS reg;
    reg.h.ah = 3;
    reg.h.bh = 0;
    int86(0x10,&reg,&reg);
    reg.h.ah = 9;
    reg.h.al = character;
    reg.h.bl = color;
    reg.x.cx = 1;
    int86(0x10,&reg,&reg);
    reg.h.ah = 2;
    reg.h.dl = reg.h.dl+1;
    int86(0x10,&reg,&reg);
}

/*
read_char_from_screen = reads a character from the screen into 'ch'
*/

char read_char_from_screen()
{
    char ch;
    union REGS reg;

    reg.h.ah = 3;
    reg.h.bh = 0;
    int86(0x10,&reg,&reg);
}
```


When this key is hit, it will immediately cause an exit from the screen mode of operation. The next parameter, called *first_line_loc*, determines the line on which the first of the two menu lines begins. It may be any screen line from 0 to 23. Normally the menu lines should be either at the top or bottom of the screen.

The final parameter is the address of the bit map, which determines which are the permissible positions of the cursor. The bit map has already been described above.

Supporting Functions

The menu function uses several supporting functions. These include *clearscreen*, which clears the screen by filling it with spaces of a designated color; *gotoxy*, which positions the cursor at a desired column and row; *put_colorchar*, which displays a character at the cursor location with a specified color combination and moves the cursor to the next column; *color_printf*, which acts like the standard C *printf* function except that it displays its data with a selected color combination; and *change_line_color*, which changes the color of a line of characters from the current cursor position up until a double space is encountered. Listings of these functions are shown for completeness. Many of them may already be available in standard libraries, but they are not included with the current version of Turbo C. If you are adapting the menu function for a monochrome display, standard monochrome equivalent functions may be used in place of some of these functions.

Conclusions

The menu function provides a great deal of flexibility in manipulating data and making menu selections. About the only restriction is that all menu function item names must fit on one line. Colors, instructions, titles, order of mode, cursor settings, and whether or not alphanumeric characters are to be displayed are all under the control of the programmer through the manner in which he calls the menu function. □

Listing 1 — Cont'd

```
reg.h.ah = 8;
int86(0x10,&reg,&reg);
ch = reg.h.al;
attr = reg.h.ah;
reg.h.ah = 2;
reg.h.dl = reg.h.dl+1;
int86(0x10,&reg,&reg);
return ch;
}

/*
set_cursor = sets up array of permissible cursor positions
*/

void set_cursor()
{
    int i,j,indx,remainder,key_value;
    char interim,map[25][10];

    FILE *f1;
    f1 = fopen("matrix.c","w");

    for(i=0;i<=24;i++)
    {
        for(j=0;j<=9;j++)
        {
            map[i][j]=0x00;
        }
    }
    row=0;
    column = 0;
    gotoxy(column,row);
    while ((key_value = getch()) != 13)
    {
        if (key_value == 0)
```



The User Interface Library for C

Create FAST & COMPACT applications in C

VLIB is a comprehensive library of over 300 easy to use C functions for building sophisticated PC applications.

- ◆ Menus ◆ Forms ◆ Memo Editor ◆
- ◆ Mouse Support ◆ Pop-up Messages ◆ Pick Lists ◆
- ◆ Dialog Boxes ◆ Colors ◆ Windows ◆ Buttons ◆
- and much more!

VLIB produces the *smallest, fastest* programs of *any* C library! For *Microsoft C* and *Quick C*, *Borland Turbo C* and *C++*. All memory models. No royalties! Free phone support! 30 day money back guarantee!

Free Demo Disk

Call (408) 984-2256

Pathfinder Associates

291 Madrone Ave., Santa Clara, CA 95051
FAX (408) 244-5665 BBS (408) 246-0164

NEW! Version 4
with FLEX functions!

Only \$189

includes libraries, manual,
and full source code.

◆ Request 165 on Reader Service Card ◆

C/C++ LIBRARIES

*More effective and easier to use
than standard class libraries.
Prevents spaghetti++.*

DATA STRUCTURE LIBRARY FOR C

Lets you code and debug three times faster with linked lists, arrays and hash tables, just like you use strcmp() today. Data pointers disappear from your code. One command moves data to disk. You'll have fully-typed, generic, persistent data you thought was possible only in C++.

C++ CLASS LIBRARY

Prevents spaghetti++ on large projects. It runs faster and uses less memory than templates, and makes standard class libraries obsolete. It separates objects from relations. Automatically generated classes prevent build-up of hierarchies. Associations map directly into your code.

DATABASE MAKER

Creates instant memory resident databases that run within your C++ code, and are much faster than expensive OODB systems. Database schema uses general data structures, and is not limited to a specific language like SQL. Very fast disk I/O. Schema migration is supported.

**Lets a novice code like a pro.
Turns a pro into a wizard.**

- fast professional tool for commercial use
- improves maintenance and debugging
- data integrity, no dangling pointers
- highly efficient, no run-time overhead
- easy to use, one day will get you started
- automatic persistence (binary/ASCII)
- no custom I/O functions, structure blasting
- portable data: C/C++, UNIX/DOS
- memory management, external allocators
- interactive data browser
- works with debuggers and other tools
- on-line help, extensive manuals
- full source, self-testing suite, examples
- industry proven, three years in use
- UNIX (SUN, HP, IBM, etc.), DOS, Windows, Macintosh, and most C/C++ compilers

Large royalty-free library

Linked lists, trees, graphs, collections, dynamic arrays, hash tables, stacks, ER models, binary heaps, time stamp, LISP-like properties, disk pager, and many other objects.

Prices: DOS/MAC \$199 (full source, all three products combined), UNIX Workstation or DOS Site License \$495 each. Specify 1.2/1.44 DOS disks or SUN tar tape. Overseas add \$25. Prices in US\$, VISA, MC, PO, or cheque.

613-838-4829 fax: 838-3316

\$199 Full Source



Code Farms Inc.

7214 Jock Trail, Richmond,
Ontario, K0A 2Z0, Canada
613-838-4829 fax: 838-3316

Listing 1 — Cont'd

```

        key_value = getch()+128;
switch(key_value)
{
    case 8: /*Backspace*/
    case 203: /*Left Arrow*/
        --column;
        break;
    case 205: /*Right Arrow*/
        ++column;
        break;
    case 208: /*Down Arrow*/
        ++row;
        break;
    case 200: /*Up Arrow*/
        --row;
        break;
    default:
        putch(key_value);
        column++;
}

if (column > 79)
{
    column = 0;
    row++;
}
if (column < 0)
{
    column = 0;
    row--;
}
if (row < 0)
    row = 0;
gotoxy(column,row);
}

row=0;
column = 0;
while (row*column < 1896)
{
    gotoxy(column,row);
    ch = read_char_from_screen();
    if ((ch == 'x') || (ch == 'X'))
    {
        indx = column/8;
        remainder = column - indx * 8;
        interim = 0x01;
        interim = interim << remainder;
        map[row][indx] = map[row][indx] | interim;
    }
    column++;
    if (column > 79)
    {
        column = 0;
        row++;
    }
    gotoxy(column,row);
}

clearscreen(30);
fputc(' ',f1);
gotoxy(0,0);
for (i=0;i<24;i++)
{
    for (j=0;j<9;j++)
    {
        fprintf(f1,"0x%x",map[i][j]);
        if ((i != 24) || (j != 9))
            fputc(' ',f1);
    }
    fprintf(f1,"\n");
}
fputc('}',f1);
fclose(f1);
}

/* End of File */

```




INSTALL.BAT

FIRST IMPRESSIONS LAST.

INSTALL 3.1 GIVES YOUR SOFTWARE PRODUCT A PROFESSIONAL INTRODUCTION.

Installation procedures that rely on batch files and DOS commands not only look amateurish but also have limited error handling capabilities. Today's users expect quality and ease of use from their software beginning the moment they open the package. A smooth, professional installation makes an important first impression.

NEW FEATURES

Over 50 new/enhanced keywords
Now licensed once, like a compiler
Supports products of unlimited size
INSTALL.EXE is still less than 80K
Displays no copyright banner
Supports Borland C++ 3.0

Microsoft Windows Version
coming soon!

PROVEN RELIABILITY

INSTALL 3.1 has been used for over four years by some of the biggest (and smallest!) names in the industry, in the U.S. and abroad, to install millions of copies of their programs. Add your name to that list today.

INSTALL PRO

- Builds Distribution Disk Sets Automatically
- Full Screen Interface
- Creates Script Files and DISK.ID Files
- High Performance Data Compression
- Slashes Work Time Up to 75%
- Splits & Compresses Files Of Any Size
- Can Tag A File, Directory, Or Directory Tree With One Keystroke
- Can Automatically Format 360K, 720K, 1.2M, and 1.44M Disks
- Fully Compatible With Install 2.0 & Later

KNOWLEDGE DYNAMICS CORPORATION

P.O. Box 1558
Canyon Lake, Texas 78130-1558

CALL OR FAX BY 10 am (CST) AND RECEIVE INSTALL 3.1 TOMORROW

◆ Request 453 on Reader Service Card ◆

EVERYTHING YOU NEED

INSTALL 3.1 is customized for your product with an ASCII text file called a "script file". INSTALL 3.1 comes with many sample script files that you can modify and use immediately. No programming is necessary to create most installations. However, C source code is included for your convenience and technical support is free.

FAST AND SIMPLE

Use INSTALL 3.1 to create an elegant installation procedure that will increase users' confidence in your product. INSTALL 3.1 takes advantage of all available RAM for lightning-fast file transfers. All your documentation has to tell users is TYPE "A:INSTALL".

30 DAY MONEY-BACK GUARANTEE

Free technical support. No royalties. MasterCard/VISA/COD/POs welcome.

\$399.95 INSTALL 3.1 PRO
\$249.95 INSTALL 3.1 Standard
\$ 99.95 International Option
\$ 99.95 OS/2 Option

SALES 1/800-331-2783X195

International 1/512-964-3994
24 hour FAX 1/512-964-3958
24 hour BBS 1/512-964-3929

Yet Another C++

Adolfo Di Mare

I use C to program information systems that commonly handle money quantities. Using *doubles* to represent money quantities can fail because the decimal digits are not represented correctly and round-off errors sometimes occur. A friend of mine, who uses Canon BASIC, has always used *floor(double*100.0)* to represent money quantities. Any amount is represented as a *double* with no fractional part. This trick loses no decimals because it

uses the *double* type, with its 15+ digit precision, as a compiler supported *long long*.

For example, \$25.35 is represented as the *double* 2535.0, using a scale factor of 100 (for two decimal places). The problem I kept facing in my C programs was remembering when to multiply by 100 and when not to. For example, adding two *doubles* that represent money quantities doesn't require you to multiply by 100

```
$25.35 + $35.75
-> 2535.0 + 3575.0 == 6110.0
-> $61.10
```

In addition, one should never multiply by the scale factor when multiplying a money quantity

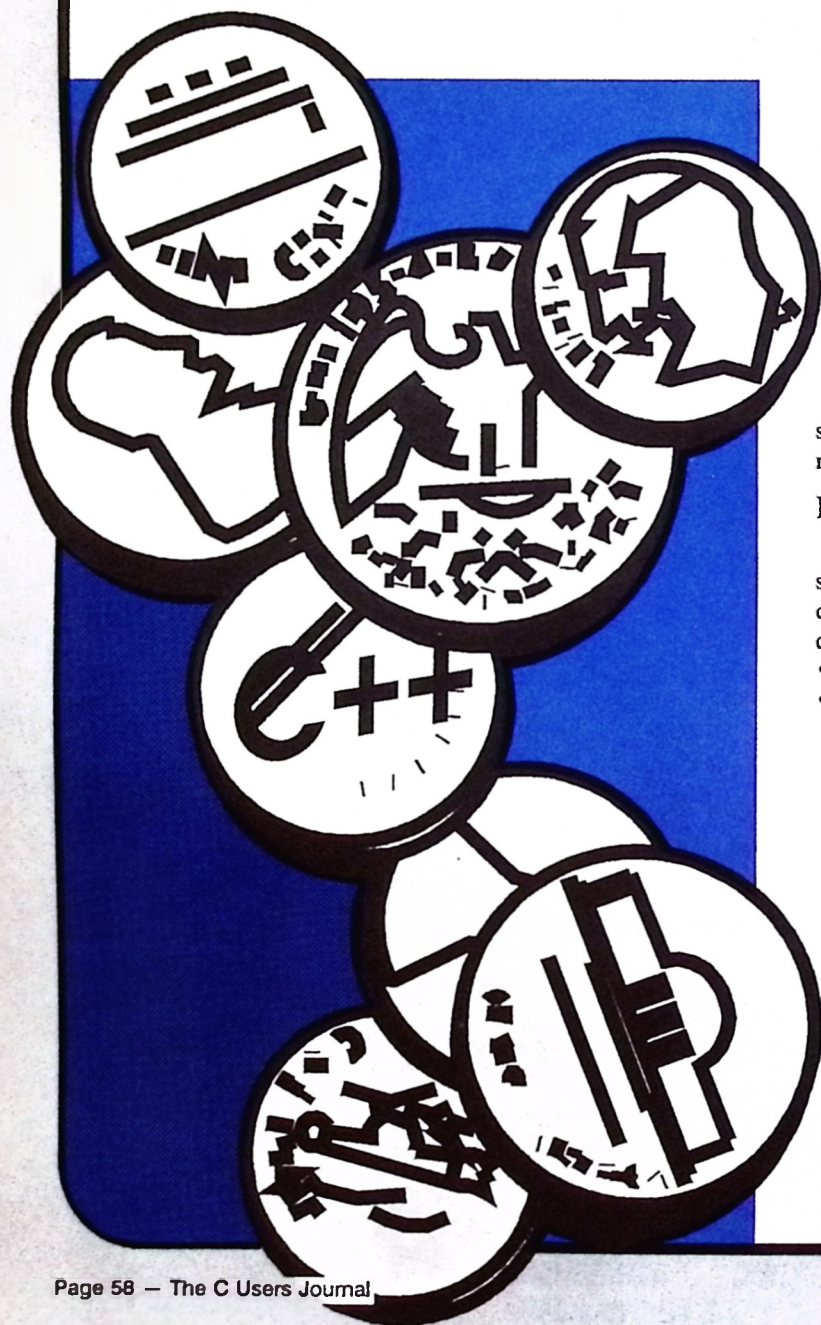
```
$100 * (1.0+0.06)
-> 10000.0 * (1.06) == 10600.0
-> $106
```

It is very easy to forget to multiply by the scale factor when using *doubles* as money. When I started programming in C++, I realized that a C++ money class was the solution for these problems.

Defining Requirements

After examining Zortech's C++ *money* class (see the sidebar on available tools), I sat down to define the requirements for my own *money* class. I came up with seven design goals:

- Money quantities should behave as regular numbers.
- The *money* class should be portable and not compiler vendor dependent.
- The programmer should be protected from misusing money quantities.
- It should be possible to use standard library functions with money quantities.
 - Most operators should be inline, to let the compiler optimize the generated code.
- The money header file should be short.
- The programmer should be able to define the number of decimals in a money data item.



Money Class

I ended up having to devote more storage space to the *money* variable than I originally wanted: eight bytes for most computers, compared to six (25 percent less) for the Zortech implementation.

Listing 1 is the header file *money.h*, which defines and implements the *money* class. All the methods in this class are inline, fulfilling my fifth requirement. The various arithmetic operators cater to the diverse situations found in real-life programs.

The *money* class permits a programmer to write expressions such as

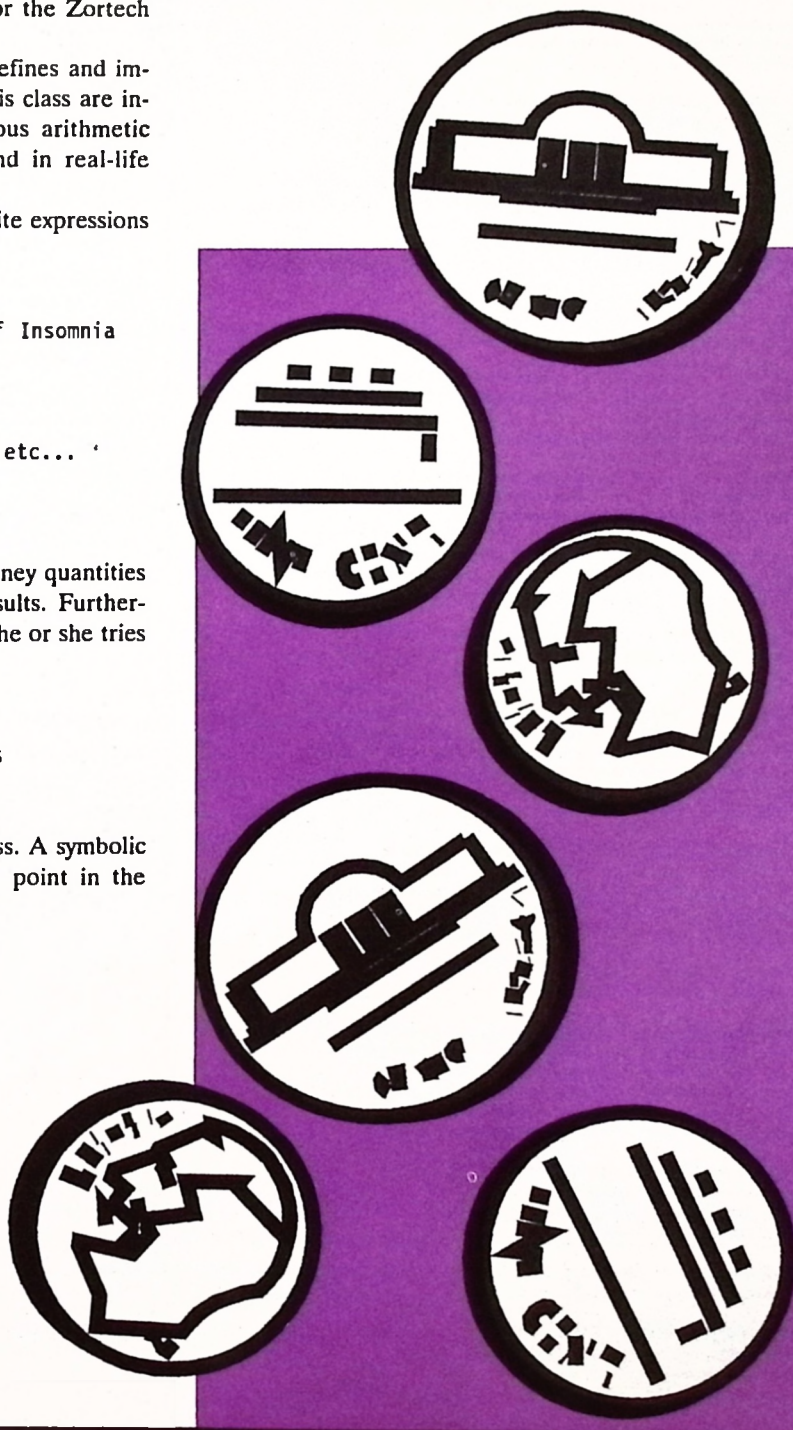
```
money mm,m = 1000; // I've got a thou'
double tax = 0.23; // State Government of Insomnia
m *= (1-tax); // This is what I have
m = m+500; // Thanks, Mom...
mm = 500;
m = m + 1500 - (mm / m) * (1.0/3.0); // etc...
printf("Salary = %10.2f\n",
      (double) ((10+mm)/m * m));
```

In a nutshell, the programmer can freely mix money quantities with regular numbers to obtain the correct results. Furthermore, the compiler will warn the programmer if he or she tries to misuse money data items, as in

```
money m, mm; // ok
double d = m*mm; // can't multiply moneys
mm = d/m; // can't divide by money
```

Listing 2 is a test program for the *money* class. A symbolic debugger will illuminate the goings-on at each point in the program.

Adolfo DiMare preaches and does research in programming at the Universidad de Costa Rica. When he is not busy reading The C Users Journal, he can be reached at (506) 24-0504 or as adimare@OCRVM2 through BITNET.



Listing 1 (money.h)

```

/* @(#) money.h          Copymiddle 1991 Adolfo Ol Mare */
/*
/*      Yet Another Money C++ Class
/*
/*      Use freely but acknowledge author and publication.
/*      DO NOT SELL IT. The author reserves all rights!
*/

/*      BITNET: adimare@UCRVM2 */

/* Compiler:      Borland C++ v 2.0 */
/* [should work with Turbo C++] */

#ifndef _money_h
#define _money_h

extern "C" { // avoid type clashes with the C library
#include <math.h> /* floor() */
#include <float.h> /* DBL_DIG */
}

#ifndef MONEY_DECIMALS /* number of decimals in */
#define MONEY_DECIMALS 2 /* any money quantity */
#endif /* don't use parenthesis! */

#define VAL(n) n /* 1 level indirection */
#define TENPOW(n) _VAL(1.0e##n) /* Trick to yield 10^n */

#define MONEY_DIG DBL_DIG

class money {
public:

    static int decimals() { return MONEY_DECIMALS; }
    static int digits() { return MONEY_DIG; }
    static double SCALE()
    { return TENPOW(MONEY_DECIMALS); }

    money(); // do nothing constructor
    money(double); // constructor from double
    money(const money&); // copy constructor

    money& operator= (const money&); // copy operator
    money& operator= (double); // copy from double
    operator double() const; // convert to double

    int OK() const; // check money's invariant
    void FIX(); // get rid of unwanted decimals

    friend money operator + (const money&, const money&);
    friend money operator + (double, const money&);
    friend money operator + (const money&, double);
    friend money operator - (const money&, const money&);
    friend money operator - (double, const money&);
    friend money operator - (const money&, double);

    friend money operator* (const money&, double);
    friend money operator* (double, const money&);
    friend double operator/ (const money&, const money&);
    friend money operator/ (const money&, double);
    friend money operator% (const money&, const money&);

    // money * money is NOT valid
    // double / money is INVALID

    friend int operator == (const money&, const money&);
    friend int operator != (const money&, const money&);
    friend int operator < (const money&, const money&);
    friend int operator > (const money&, const money&);
    friend int operator <= (const money&, const money&);
    friend int operator >= (const money&, const money&);

    money& operator += (const money&);
    money& operator += (double);
    money& operator -= (const money&);
    money& operator -= (double);

    money& operator *= (double);
    money& operator /= (double);

    friend money operator+ (const money&);
    friend money operator- (const money&);
    money& operator++();
    money& operator--();

    friend int operator! (const money&);

    friend money abs(const money&);
    friend money flatten(
        const money& m,
        double cents=0.25, int rounding = 1 /* TRUE */);

protected: // let users change the class behaviour
    double m_money;
};

// Constructors && assignment
inline money::money() {
// do nothing constructor, for efficiency
}
inline money::money(double d) {
// construct from double
    m_money = d*SCALE();
    FIX();
}
inline money::money(const money& m) {
// copy constructor
    m_money = m.m_money;
}

inline money& money::operator= (const money& m) {
// copy operator
    m_money = m.m_money;
    return *this;
}
inline money& money::operator= (double d) {
// assign from double
    m_money = d*SCALE();
    FIX();
    return *this;
}
inline money::operator double() const {
// convert to double
    return m_money / SCALE();
}

inline int money::OK() const {
// Returns TRUE (1) when the quantity stored
// in *this really corresponds to a money
// quantity.

    money temp;
    temp.m_money = m_money;
    temp.FIX();
    return (
        (temp.m_money == m_money)
        &&
        (fabs(m_money) < (TENPOW(DBL_DIG) / SCALE()))
    );
}

inline void money::FIX() {
// Deletes all decimals digits beyond
// the MONEY_DECIMALS decimal place.
// - If the value is out of range, FIX
// won't fix it.
    m_money =
        (m_money > 0.0
         ?
         floor(
             m_money
             #ifdef MONEY_ROUNDING
             + 0.5 // 0.49 is also an option...
             #endif
         )
        :
        ceil(
            m_money
            #ifdef MONEY_ROUNDING
            - 0.5
            #endif
        )
    );
}

// add
inline money operator+ (const money& m, const money& nm) {
    money temp; // don't mult*SCALE()
    temp.m_money = m.m_money + nm.m_money;
    return temp;
}

```


Implementation Details

I first tried to fake money quantities using *longs*, but the difficulty in doing so stopped me from pursuing this approach. (The Zortech tool's *money* type is implemented using integer arithmetic.)

After I realized that the *money* variable would be a *double*, the main implementation problem was to keep track of when to multiply by the scale factor and when not to. This I accomplished in each of the overloaded arithmetic operators.

I also implemented the *money::FIX()* member functions to get rid of the excess decimals whenever appropriate. When the preprocessor constant *MONEY_ROUNDING* is defined, the excess decimals in a *double* assigned to a *money* variable are rounded. Otherwise the excess decimals are truncated

```
money m(1.5199); // $1.52, when
                  // MONEY_ROUNDING
```

```
money m(1.5199); // $1.51, when
                  // not MONEY_ROUNDING
```

The programmer cannot selectively choose whether to round or not case by case because the decision is made at compile time.

The preprocessor constant *MONEY_DECIMALS* defines how many decimals a money item has. The member function *money::SCALE()* is implemented using a preprocessor trick that returns the scale factor used to multiply a *double* to make it a money quantity. If three decimals are needed for money items, the scale factor would be $1,000 = 10^3$. In some countries the inflation is so high that the number of decimals is negative. In this case the scale factor would be a number less than one. Since *money::SCALE()* is an inline function, the compiler can optimize out the division by the scale factor in some cases. If the programmer doesn't define *MONEY_DECIMALS*, then the class uses a default value of two decimals. In my programs, I define *MONEY_DECIMALS* before including the *money.h* file

```
#define MONEY_DECIMALS 4 // must
                          // use a decimal number
#include "money.h" // or TENPOW
                  // bombs.
```

The vector constructor *money::money()* does not initialize a money item, because in many cases doing so would be wasteful. The programmer can consider money items as regular numbers. The compiler should be able to optimize out this constructor if it is used.

Though the class *money* implements most arithmetic operators, it does not implement the following

```
money operator* (const money&,
                 const money&),
money operator/ (const double,
                 const money&);
```

It just does not make sense to use these operators in a program. If you use them, you will get a compile time error (a dissatisfied programmer could add them to the class easily).

As arithmetic operators in *money.h* demonstrate, the class is programmed to minimize the number of times that each *double* must be scaled up by the scale factor.

The comparison operators are defined only for money items. Thus when a money data type is compared to a *double*, the compiler promotes the *double* to a *money* variable using the constructor *money::money(double)*. To prevent the promotion, the programmer should use an explicit typecast

```
double d = 15.253; // 15.253
money m = 15.25; // $ 15.25

if (d == m) { // TRUE: d
              // becomes money(d)
}
if (d == (double) m) { // FALSE:
                       // 15.253 != 15.25
}
```

The function *flatten(money, cents, rounding)* rounds up a money quantity to the nearest value that can be paid in coins. For example, Costa Rica has no one cent coin because the smallest coin,



The "Standard" Foundation C++ Class Library

NOW HAS TEMPLATES!

Tools.h++

At Rogue Wave we are committed to allowing our users to tap all of the power that C++ offers. For today's compilers, that means templates. *Tools.h++*, our best-selling industry standard C++ library, now includes *Templates*. *Tools.h++* is a complete toolbox of over 60 C++ classes. It is a set of efficient and variable C++ foundation classes that will make virtually any programming job easier.

- Time and Date handling and manipulation classes.
- String and Character manipulation classes.
- Singly and Doubly linked lists, Stacks, Queues and Vectors classes.
- Smalltalk™-like Collection classes: Set, Bag, SortedCollection, OrderedCollection, Dictionary, Stack, Queue, etc.
- Regular Expression Class for search and replace.
- Tokenizer Class for easy string parsing.
- File Class to handle file manipulation with read, write, seek, erase, etc.
- B-tree Class to handle efficient keyed access of disk records.
- File Space Manager Class to allocate, deallocate and coalesce space within files.
- Utility to detect memory leaks, reuse after deletion and wild pointers.
- Virtual and Buffered Page Heap to manage objects bigger than 64k.
- Other classes include: Bit vectors, Virtual I/O streams, Cache managers, Error handling and many more!

Rogue Wave's *Tools.h++* is an "industrial strength" library. All classes have not been derived from a single root object, so they can be easily integrated with other class libraries. All classes have a Persistent Store facility that allows complex objects to be stored and retrieved on heterogeneous networks or the Windows 3.0's Dynamic Data Exchange or clipboard facility. The classes are bulletproof, optimized for speed, and well tested. *Tools.h++* is an excellent example of how to write true C++ code correctly. *Source Code* is included.

Compatible with
MS-DOS, UNIX,
Windows 3.x, OS/2
and other OS's.

Linpack.h++

The full power of the widely used Fortran LINPACK routines plus much more, completely recoded in a True object-oriented C++ interface. Fully optimized with assembly language BLA routines: includes assembly language BLA's for 80x86 & 80x87 processors. Includes *Math.h++*, the fastest C++ Math library available. Complete vector and matrix classes - complex, double, int, char, float, etc. Statistics—Random number generators (Gaussian, Poisson, etc.); histograms; probability functions; linear regressions, etc. Linear Algebra—matrix inversions; LU decomposition; solutions of linear equations; determinants. Signal Processing—Fast Fourier Transforms: complex; real; 1- and 2-D. Slice and pick operations; SubMatrices.

Full Source Code!

- Complete manual full of C++ hints and tips
- Numerous examples
- No Royalties
- Compiles with Borland C++, Zortech, Oregon, Lint, Microsoft C++ & most other C++ compilers

CALL NOW: 1-800-487-3217



ROGUE WAVE
SOFTWARE

P.O. Box 2328 • Corvallis, OR 97339
503-754-3010 • FAX 503-757-6650

◆ Request 124 on Reader Service Card ◆

Listing 1 — Cont'd

```

inline money operator+ (double d, const money& m) {
    return (money(d)*m);
}
inline money operator+ (const money& m, double d) {
    return (m*money(d));
}

// subtract
inline money operator- (const money& m, const money& mm) {
    money temp;
    temp.m_money = m.m_money - mm.m_money;
    return temp;
}
inline money operator- (double d, const money& m) {
    return (money(d)-m);
}
inline money operator- (const money& m, double d) {
    return (m-money(d));
}

// multiply
inline money operator* (const money& m, double d) {
    money temp;
    temp.m_money = m.m_money * d; // don't mult by SCALE()
    temp.FIX(); // this could be delayed...
    return temp;
}
inline money operator* (double d, const money& m) {
    return (m*d);
}

// divide
inline double operator/ (const money& m, const money& mm) {
    return m.m_money / mm.m_money;
}
inline money operator/ (const money& m, double d) {
    money temp;
    temp.m_money = m.m_money / d;
    temp.FIX(); // this could be delayed...
    return temp;
}
inline money operator% (const money& m, const money& mm) {
    money temp;
    temp.m_money = fmod(m.m_money, mm.m_money);
    temp.FIX(); // this could be delayed...
    return temp;
}

// compare
inline int operator == (const money& m, const money& mm) {
    return m.m_money == mm.m_money;
}
inline int operator != (const money& m, const money& mm) {
    return m.m_money != mm.m_money;
}
inline int operator < (const money& m, const money& mm) {
    return m.m_money < mm.m_money;
}
inline int operator > (const money& m, const money& mm) {
    return m.m_money > mm.m_money;
}
inline int operator <= (const money& m, const money& mm) {
    return m.m_money <= mm.m_money;
}
inline int operator >= (const money& m, const money& mm) {
    return m.m_money >= mm.m_money;
}

inline money& money::operator += (const money& m) {
    m_money += m.m_money;
    return *this;
}
inline money& money::operator += (double d) {
    m_money += d*SCALE();
    FIX();
    return *this;
}

inline money& money::operator -= (const money& m) {
    m_money -= m.m_money;
    return *this;
}
inline money& money::operator -= (double d) {
    m_money -= d*SCALE();
    FIX();
    return *this;
}
inline money& money::operator *= (double d) {
    m_money *= d;
    FIX();
    return *this;
}
inline money& money::operator /= (double d) {
    m_money /= d;
    FIX();
    return *this;
}

// unary op's
inline money operator+(const money& m) {
    return m;
}
inline money operator-(const money& m) {
    money temp;
    temp.m_money = -m.m_money;
    return temp;
}
inline money& money::operator++() {
    m_money += SCALE();
    #if (MONEY_DECIMALS<0)
        FIX(); // avoid problems because of
    #endif // the representation of 10^-n
    return *this;
}
inline money& money::operator--() {
    m_money -= SCALE();
    #if (MONEY_DECIMALS<0)
        FIX();
    #endif
    return *this;
}
inline int operator!(const money& m) {
    return m.m_money == 0.0;
}

inline money abs(const money& m) {
    money temp;
    temp.m_money = fabs(m.m_money);
    return temp;
}

money flatten(const money& m, double cents, int rounding) {
    // Returns a money data item where the cents are
    // rounded modulo "cents". In this way cents can
    // be stripped of money items when the currency
    // does not have all the coins required to pay
    // every possible quantity.
    money temp;
    double c = floor(fabs(cents*money::SCALE())); // cents
    double r = fmod(m.m_money, c); // remainder
    temp.m_money =
        (rounding || (2.0 * r <= c)
         ? m.m_money - r
         : m.m_money - r + c
        );
    return temp;
}

/* Avoid name space overcrowding */
#undef VAL
#undef TENPOW /* jic: Just In Case! */

#endif /* _money_h */

/* End of File */

```


called a *peseta*, is worth 25 cents of a *colón*. In the following example the money item *m* is rounded up to *pesetas*

```
money m = 125.80; // 125.88 colones
money mm = flatten(m); // 125.75 colones
```

The header file called *money.h* implements the complete *money* class using inline functions, which permit the compiler to optimize out any floating point operations it can. I decided not to provide more operators for money items, because the type converter *money::operator double()* allows the standard math functions to work with money items. *money.h* includes only two files, *math.h* and *float.h*, which contain the prototypes for the functions *floor()*, *ceil()*, *fmod()*, and the constant *DBL_DIG*. In order to use bigger money quantities, you can declare every *double* variable in *money.h* as a *long double*.

As implemented, the money class should be quite portable because it does not make use of any odd C++ constructs. Depending on the compiler used, it is quite possible to optimize out many of the inline operators, and thus yield efficient programs.

Conclusion

This tiny C++ money class lets the programmer use money items with ease. The implementation is as efficient as using floating point values in arithmetic expressions. This should be reason enough to use it right away. □

Listing 2 (money.c)

```
/* 0(0) money.c                                1991 Adolfo Di Mare */
/*                                              adimare@UCRVH2 */
/*                                              */
/* Test driver for money.h                      */
/*                                              */
/* Compiler:                                Borland C++ v 2.0 */
/*                                              [should work with Turbo C++] */

/*
To see what is going on, you need to use your symbolic
debugger to examine each of the declared variables.

For Borland C++, I used the following watches:

m_money    d,f18
m,r        i
mm,r       1
elapsed    s,f18

Change the compile time macros to see how money's
change their behaviour.
*/

#define TEST

#ifdef TEST /* ( */

if 0
#define MONEY_ROUNDING /* Force rounding of doubles */
#endif

#define MONEY_DECIMALS 2 /* 2 decimals for money data */

extern "C++" {
#include "money.h"
```

CHOICE INSTALL

Easy to use, Efficient Installation program with:

- All of the most needed features and more.
- Automatic detection of system features.
- Runs under DOS, Windows, or OS/2 comp. box.
- No script language to learn.
- Fast file transfers using all available RAM.
- Uses Microsoft, Borland, Zortech C, C++.
- Takes less than 100K on your distribution disk.

No programming! Run the generator, "fill in the blanks", compile and link, run the build program to create your disks and you're ready to ship!

90 day money back guarantee, no royalties or extra licenses for multiple products. Send \$149, FAX or call (602)298-0666 with Visa or M/C. Fully functional demo disk: \$5, credited on purchase. Specify disk size (3.5" or 5.25" HD).

ChoiceWare

8802 East Broadway Suite 211, Tucson, AZ 85710

Microsoft, Borland, Zortech are trademarks of their respective companies.

◆ Request 362 on Reader Service Card ◆

BBROWSE

Extends BRIEF & PVCS (Sage) to include a Multi-language Source Browser!

- ◆ **Languages Supported:** Microsoft C 6.0, MASM 6.0, BASIC 7.1, COBOL 4.0, FORTRAN 5.1 & later versions.
 - ◆ **No Waiting:** Browser database generated DURING compilation—no time consuming scanning pass!
 - ◆ **NOT A TSR:** BBROWSE is an extension of your editor, it doesn't take up valuable memory space.
 - ◆ **Project-wide Symbol Usage:** The Symbol Details Report shows how a particular symbol is used throughout a multi-language project.
 - ◆ **Hypertext Editing:** A single keypress brings up your source file for editing, with your cursor at the position of the definition or reference.
 - ◆ **Find Dead Wood:** The Unused Symbols Report finds abandoned functions and variables.
 - ◆ **Generate Documentation:** All reports can be printed to satisfy your documentation needs.
 - ◆ **Accuracy:** BBROWSE uses a Microsoft generated database. You get an EXACT code representation, NOT just an interpretation from a generic source code scanner.
- Requires a Microsoft Professional Language and BRIEF 3.0+ or PPE 2.0+

BBROWSE: only \$99!

1-800-453-5277

Free
Demo Disk
Available

THE SOFTWARE ANNEX
5145 W. 12th Avenue Court
Boulder, CO 80504
Voice: 303-445-5277

◆ Request 109 on Reader Service Card ◆

Listing 2 — Cont'd

[illegible]

```
// $10 == 0L * $77 + [$10]

m = 77; // $ 77.00
nm = 10; // $ 10.00
m = m % nm; // $ 7.00
// $77 == 7L * $10 + [$7]

m++; // $ 8.00
m--; // $ 7.00

m = 11.75; // $ 11.75
m += 0.12; // $ 11.87
nm = flatten(m,0.25,1); // $ 11.75

m += 0.01; // $ 11.88
nm = flatten(m,0.25,1); // $ 12.00

m = 11.75; // $ 11.75
m += 0.12; // $ 11.87
nm = flatten(m,0.25,0); // $ 11.75

m += 0.01; // $ 11.88
nm = flatten(m,0.25,0); // $ 11.75

m -= 5;
m += 0.12; // $ 7.00
if (m == 0 || 0 == m) { // nep
    m += d;
} else if (!m == m) { // nep
    m = m;
} else if (m > m) { // nep
    m = m;
} else if (m < m) { // nep
    m = m;
} else if (m != m) { // nep
    m = m;
} else if (m >= m) { // yep
    m *= 11; // $ 77.00
    m += 15; // $ 92.00
}
l = i = m; // 92
m = -m; // $ -92.00
nm = i * l; // $ 8,464.00
m = m % nm; // $ -92.00

d = 15.253; // 15.253
m = 15.25; // $ 15.25

if (d == m) { // TRUE: d becomes money(d)
    l = 0; // l = 0L
}
if (d == (double) m) { // FALSE: 15.253 != 15.25
    l++; // l == 0L
}

// simulate a TAX calculation
m = 0.0;
for (i = 1; i <= 100; i++) {
    d = i * 1.005; // 0.05% tax
    m += d;
} // $ 5,075.00
nm = 100; // $ 100.00
m /= (double) nm; // $ 50.75
m /= 3; // $ 16.91
m *= 3; // $ 50.73

nm = nm / nm; // 1.00
d = m * (m / nm); // 2573.530000000000002

d = 1.0 / 3.0 * m; // 16.910000000000000001
nm = 1.0 / 3.0 * m; // $ 16.91
```


Available Tools

My first move in creating a *money* class was to examine the available tools to handle money quantities. I looked at the following:

- Borland's C++ BCD class
- Zortech's C++ BCD class
- Zortech's C++ money class

BCD stands for *Binary Coded Decimal*.

The binary representation used in computers is a sequence of binary "digits" called *bits*. BCD, on the other hand, stores a number's value in base 10 digits, usually using their binary values. For instance, the number 1234 is stored in a pair of bytes using the same bit pattern as the *0x1234* hexadecimal constant. Since most computers use eight-bit bytes, each byte will hold two BCD digits. Many bit patterns are not valid when interpreted as BCD numbers: *0xFFFF*, *0x0A0A*, *0x123A*, are all invalid BCD quantities because in decimal notation we can use only the digits 0...9.

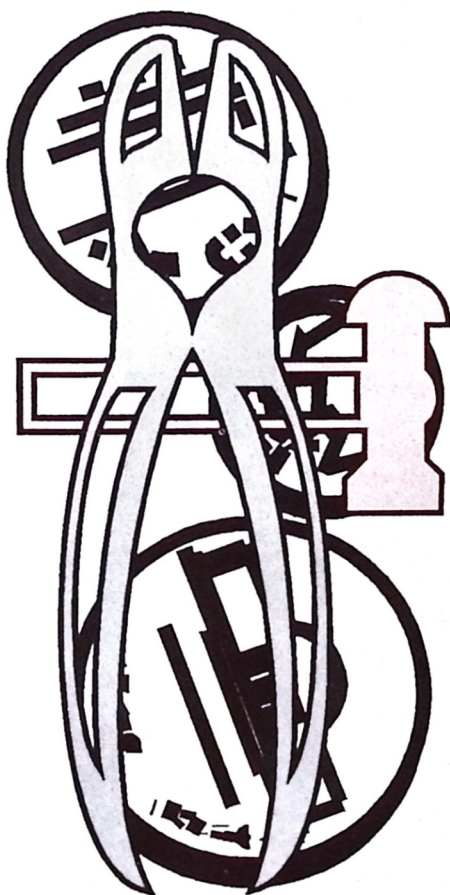
I chose not to use Borland's BCD class because it does not provide specific support to handle money quantities. For example, whenever you multiply and divide two money variables, you must figure out what to do with the digits that go beyond the number of cents. The header file in Borland C++ v2.0 pulls in the *iostream.h* header file, which slows down compilation quite a bit. I still don't use streams, and I don't like to be forced to use them if I don't have to. Also, much of the BCD library must be loaded when using BCDs, increasing the size of executable files even more. Finally, Borland doesn't include the source code for this class with the compiler, though it is available at extra charge. I have learned to examine a library's source code before using it.

Zortech's BCD class shares most of these inconveniences.

The Zortech C++ Tools package implements a *money* class, where a money quantity is a two member structure:

```
class money
{ long dollars;
  int cents; // ... };
```

In most implementations, a *long* can hold only nine or ten decimal digits. The range of numbers is limited compared to that granted by *doubles*. This class is efficient because the operators are implemented using integer arithmetic and each money variable is quite short. □



PROGRAMMER'S SHAREWARE

Toll Free Ordering 800/327-2540

Same Day Free Shipping (US only) \$10 Int'l.

- Utilities** - Most disks archived for 600K per disk
- Vaccine/Anti Virus (UT11) Protect your system from viruses!
 - Flowcharting (MS17) Programming, Organization, Electronic etc. Great!
 - POWER BATCH COMPILER (UT48)-Enhanced batch language. Fast executables.
 - Unprotect Patches (UT20) Unprotect many great business packages!
 - Archive/Unarc (UT12) Store twice as much information on a disk!
 - VGAS/VGA Graphics Toolkit (UT47)-Libraries, source code, tutorials & tools for graphics.
 - DOS Tools (UT46) - great tools and enhancements for DOS. All compressed files.

Languages

- DUKE/NUKEM (GA41)-The hottest, best animated, most fun game going!
- DOS5 Tutor (UT45)-A megabyte of great tutorial on the all new DOS 5.0
- Hyde 90 function assembler lib. (LA25). Includes complete source.
- Compiler tutorial (UT44) Write your own compiler. Full C & Turbo Pascal source.
- TurboC Tutorial (LC04/LC05) Learn Turbo C. 2 Disks Full of extensive lessons and over 60 sample programs.
- Window Boss (LC07-LC08 3 Disks) New Version! Pull-Down & Pop Up Menus, Data Entry etc. Great Documentation.
- LC15 Desmet-C-Compiler Comp. Full C-Compiler, Linker Assembler complete C lang.
- LB08 Integrated Basic Compiler. Full environment, edit, compile, debug and run all within the environment. Much like QuickBasic™!
- WP13 Multi-Edit-Editor. Awesome Programmers Editor. Register for manual documenting extensive Macro Language-THE BEST!!
- C++ Like Compiler w/ source (LC24)-OO Language w/ classes & complete C source. Examples & doc.

- LA12 Smart Disassembler, Smart Labeling. Disassemble to disk, screen, or printer.
- UT30 Programmers Ref. disk. Huge amounts of valuable Ref. material.
- (FOXPRO) 3 DISKS-Profilers, Animated Libraries, toolkits, OOP in FoxPro & more! - \$9

Hot New Shareware

- 4 DOS (UT32) VASTLY improved DOS COMMAND.COM replacement (A MUST HAVE!)
- C++ Extension (LC16/LC17 2-Disks) - Provides OOP for Microsoft™ or Turbo™ Entry etc. Great documentation. THE BEST ON THE MARKET!
- EZ-CASE (GR08) Computer Aided Software Engineering! Very Good!
- BIOS/DOS Reference (UT42) Gobs of info on mysterious internals of the BIOS and DOS. Includes a ROM capture disk util.
- TSR Library (UT43) Complete TSR development library for Pascal, C & Asem.
- FREEMACS Editor (WP27/WP28 2-Disks). Extremely comprehensive EMACS like editor, spell checker & macro lang. FULL ASEM SOURCE!
- CXL Library (LC18/LC19 2-Disks) 375 Functions in all Windows, Bar menus, Pick lists, Help systems, Video Mouse, etc.
- C++ Tutorial (LC21/LC22 2 disks) Very complete. 14 chps of text. 70 example

Language Sets

- Each disk in these sets is compressed with over 700K of utilities, toolkits, libraries, graphics & windowing support, etc. All include necessary documentation and most include source code!
- (CUB12) 12 PACKED disks (7.5 meg). A huge 260 function library. ZIP compression with source. Stream Editor, Read Pictor files more-\$30.
 - (CUB25) 25 packed disks (13meg) all of CUB12 + full database sys w/ source, ed w/ source. 375 function lib & more. \$55.

- (ASEM7) 7 PACKED disks BIOS Routines, Critical error, Communications 8087 macros, over 220 utilities with full source, and much, much more!! - \$20
- (ASEM12) 12 PACKED disks (7.5 meg). all the above plus huge macro library, complete editor w/source. TSR utilities, VGA routines, over 350 utilities - \$30
- (TPAS7) 7 PACKED disks (4.5 meg) for Turbo Pascal™. Huge function library, EGA routines, Mandelbrot, Mouse support, BTREE files, Windows, Modem. More-\$20
- (TPAS12) 12 PACKED disks (7.5 meg), all of the above plus, Menu builder, Cross Ref., Multitasking routines, OOP utilities. 100's of the files. Much more-\$30
- (C++ LIB) For Turbo C++ 12 PACKED disks (7.5 meg) Communications, Class Libraries, Mouse & Windows Toolkits, Graphics & Sprite Lib.-More-\$30
- (DBASE) 7 PACKED disks (4.5 meg) for dBASE III/IV (and clones). Fix damaged files, point & shoot menus/menu bars, dozens of applications. Much more - \$20
- (BASIO8) 8 PACKED DISKS-3 Meg-for QuickBASIC™(tm). Graphics & Math Libraries, Btree, Library with 123 functions, Windows. Much More! - \$18
- (TPOCP6) 6 disks-For Turbo Pascal 5.5/6.0. 3.5 Meg dBase appl. & util. w/ source code-\$18
- (WTN12) 12 disks (7.5 meg) Huge lib 70+ util & wallpaper, games, editors, icons & icon ed & much more. Windows 3.0 \$30
- Comp. 321 disk lib Everything & more \$550

\$3.50 per disk/10 or more, \$3.00 per disk. 3.5" format, add \$1.00 per disk. 126 Atari ST disks, as low as \$1.60 per disk.

FREE 80 PG. CATALOG

Computer Solutions/N.W.
P.O. Box 446, Benzonia, MI 49616
Info: 1-616/325-2540
MC Fax: 1-616/325-2505 Visa

◆ Request 445 on Reader Service Card ◆

Listing 2 — Cont'd

```

mm = 3 * mm / (3 * mm); // $1.00
mm = M_PI * mm / (M_PI * mm); // $1.00
mm = M_PI; // $3.14

mm = mm/mm + 1 - (3 * mm / (3 * mm)); // $1.00
m = M_E * mm / (mm * M_E) - 1; // $0.00

// m == 0.0 && mm == $1.00
for (i = 1; i <= 100; i++) {
    mm /= 3; // $ 0.33
    m = m+mm; // Add a third
    mm = 1;
} // $ 33.00

d = m; // 33.00
mm = m / 330; // $ 0.10

clock_t now;
double elapsed;

// time statistics, on an 33MHz 386
m = 0;
now = clock();
for (i = 0; i <= 100001; i++) {
    m += 1.01; // Add $1.01
}
elapsed = (clock()-now) / CLK_TCK;
d = elapsed; // 3.51 secs

m = 1;
now = clock();
for (i = 0; i <= 100001; i++) {
    m *= 1.0001; // Mult
}
elapsed = (clock()-now) / CLK_TCK;
d = elapsed; // 3.24 secs

```

```

m = 1;
mm = pow(10, 6);
now = clock();
for (i = 0; i <= 100001; i++) {
    mm = m;
    m /= 0.99001; // Div
    if (!m.OK()) {
        m.FIX(); // won't fix overflows
    }
}
elapsed = (clock()-now) / CLK_TCK;
d = elapsed; // 8.46 secs

d = m % m + 33; // 33.00
m = d; // $ 33.00
mm = d / 330; // $ 0.10

d = (10+mm)/m * m; // 10.089...

// Must use (double) type cast
printf ("Salary = %10.2f\n",
        (double) ((10+mm)/m * m));
cout << "Salary = "
    << (double) ((10+mm)/m * m) << '\n';

// valid only if you define
// ostream& operator<< (ostream&, money&)
cout << "Salary = "
    << (10+mm)/m * m << '\n';

m = d / m; // should not compile...
// m = m * m; // won't compile: AMBIGUITY ERROR!!!

#ifdef RANGER
    ranger();
#endif // RANGER

exit(0);
}

void ranger(void) {
    /*
     * Shows that indeed a double can hold up to
     * DBL_DIG digits of precision.
     */
    // This should take forever to calculate...
    char view[] = "0123456789/123456";
    double s,t;
    double tenpow,inc;
    int i;

    tenpow = pow(10, DBL_DIG); // 10^15
    inc = 100.0; // pick yours

    s = floor(tenpow+inc); // 1,000,000,000,000,000 + inc
    for (;;) {
        t = s;
        s += inc;
        i = (int) log10(t-tenpow);
        if (s == t) {
            i = (int) log10(t-tenpow);
            cout << "BOOM t = " << t << '\n';
            cout << "BOOM s = " << s << '\n';
            cout << "BOOM inc = " << s << '\n';
            cout << "BOOM i = " << i << '\n';
            view[i] = 0;
            cout << "view = " << view << '\n';
            return;
        }
    }

    return;
} // ranger

#endif /* TEST */ /* */

/* End of File */

```

INSTALIT™

for DOS • OS/2 • Microsoft® Windows
The Premier Worldwide Installation System™

Yes... you can do it with **INSTALIT™**. It has all the features you've requested over the last five years, so you won't constantly run into walls. It's capable enough to satisfy the world's most complex installation requirements, but so logical that even non-programmers can be almost immediately productive, and you can grow with it for years to come. All versions include Media-Builder™, the master diskette builder developers are calling "a godsend." No royalties or relicensing for use with additional products. You can translate all versions into additional languages without source code.

INSTALIT™ FOR WINDOWS... The Ultimate Windows Installer™

Native 3.X application weds the unmatched flexibility of **INSTALIT™** to the power of the Windows API. Generalized dialog boxes, DLL access, graphics, bitmaps and more.

SEE FOR YOURSELF!

— 30-day FREE trial! —

90-day money back guarantee!

Start today with our BBS starter kit!

BBS DEMO (205) 880-8785 or (205) 880-8782 Voice/FAX

For DOS & OS/2 - \$149

For Windows - \$199

Source Code add \$100

Specify national language

Translation pack (14+ languages) - add \$199

VISA/MC/AMEX/PO/COD accepted

Shipping \$5, next day \$15

America's Favorite Installer™
Orders only 1-800-448-4154



P.O. Box 16078 • Huntsville, AL USA • 35802

◆ Request 118 on Reader Service Card ◆

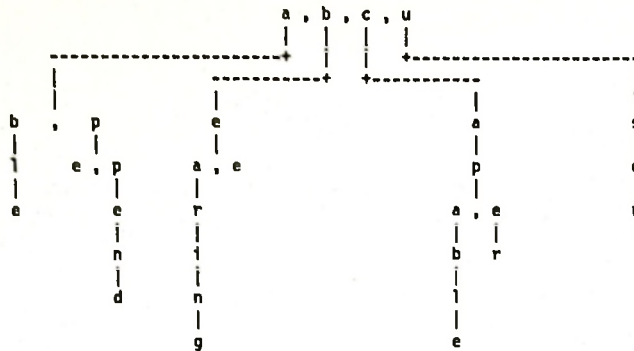
Recently, there has been a lot of talk about an old UNIX idea, that of user programmability. User programming presents some drawbacks, not least of which is the absence of a standard language. Each new program requires the user to learn a new language (unless the new program is a clone of another). Also, though different user languages may have similar syntax, they may not interpret a statement the same way. In addition, user programming languages have, until just recently, been cryptic and difficult to learn.

To create a program that is user-programmable requires that the software engineer

The first part of the system, the lexical analyzer, takes ASCII input, separates it into words, and converts those words to numeric values, called *tokens*. Words that have special meaning in the language are called *keywords*. Punctuation characters, such as ; and :, also have special meaning in the language and must be tokenized as well. The lexical analyzer determines if the input contains illegal words or punctuation characters.

The C Users Journal — Page 67

Figure 1



Example Search Trie

While the lexical analyzer breaks the input text into words, it does not determine whether the words are arranged into legal sentences. This is the job of the parser. The parser takes a stream of tokens from the lexical analyzer and attempts to determine if they form a stream of legal sentences according to the language's grammar. A grammar consists of a set of rules that describe all legal sentences possible in the language. Not all legal sentences make sense. In most programming languages, the parser will accept legal sentences that the interpreter cannot understand or execute.

Once the parser has decided that the token stream forms legal sentences, the interpreter combines the operations of semantic analysis and program execution. Semantic analysis determines what operations the program is telling the interpreter to execute. Execution is the operation of reading program tokens and translating them into a series of machine language function calls that instruct the CPU what to do. Interpreted languages execute slower than compiled languages, in part, because the interpreter must translate each program sentence into machine language every time it is executed. Sentences from compiled programs are already translated into machine language.


Lexical Analyzers And Search Tries

To facilitate the construction of lexical analyzers, I use a special class of search tree, called a *trie*. A trie is a tree data structure that allows strings with similar character prefixes to use the same prefix data and store only the tails as separate data. One character of the string is stored at each level of the tree, with the first character of the string stored at the root, and the last character of the string stored at a sub-tree node or in a leaf node. Figure 1 shows how a trie would store the following collection of words: ape, append, able, bee, bearing, cape, caper, capable, us, use and user.

Tries used for lexical analysis store token values with each character in the trie, as shown in Listing 1. Most of the token values are zero, indicating *ILLEGAL KEYWORD*. The token value of the last character of each legal word is the token value for that word. For example,

Turn your favorite C compiler into a powerful database manager with the

C/Database Toolchest



The C/Database Toolchest™ adds sophisticated file management functions to your Power C™, Turbo C®, QuickC®, or Microsoft® C compiler. With the C/Database Toolchest™, your data requires much less disk space than with programs like dBASE®, and you can access your data much faster. Of course the full power of C provides you with an unlimited amount of programming flexibility.

The C/Database Toolchest™ includes three major components:

- 1) An advanced B+tree library gives you instant access to your data.
- 2) A high-level ISAM library provides you with an easy-to-use C interface, and
- 3) A complete database manager (with C source code included) shows you how to create impressive applications.

You also receive a comprehensive 350 page manual and a utility for converting dBASE® files.

The C/Database Toolchest™ supports features that you'd expect to find only in products costing ten times as much. Advanced features include variable length records, variable length keys, multiple keys per index, and multiple indexes stored in a single file. Your data files can contain an unlimited number of records, and each record can be as large as 32K bytes in length.

About the only thing that the C/Database Toolchest™ doesn't do is cost you a lot of money. We've kept our price low so you can manage your budget as easily as your data.

Now Only \$19.95!

Order Coupon

Name _____

Street _____

City _____ State _____ Zip _____

Telephone _____

Paying By _____ Money Order _____ Check _____

Card# _____ Visa _____ MC _____ AX _____ Disc. _____

Exp. Date _____

Disk Size _____ 5 1/4" _____ 3 1/2" _____


Qty	Product	Price	Subtotal
_____	C/Database Toolchest	\$19.95	_____
_____	C/Database Library Source	\$10.00	_____
_____	Borland ISAM Library Source Code	_____	_____
Add Shipping (\$3 USA, \$20 Foreign)		_____	_____
Texas Residents Add 8% Sales Tax		_____	_____
Total Amount of Your Order		_____	_____

Order now by calling our toll free number or mail the coupon to:

Mix Software
1132 Commerce Drive
Richardson, TX 75081

1-800-333-0330

60 Day Money Back Guarantee
Not Copy Protected ■ Royalty Free
For technical support, please call:
1-214-783-6001



C/Database Toolchest and Power C are trademarks of Mix Software. QuickC and Microsoft are registered trademarks of Microsoft. Turbo C is a registered trademark of Borland. dBASE is a registered trademark of Ashton-Tate.

in Figure 1, the *struct* for the letter *e* in the trie branch under the letter *u* stores the token value for the word *USE*. The letters *s* and *r* in the same branch would have the tokens *US* and *USER* stored with them, respectively.

Using a trie in a lexical analyzer combines the operations of breaking the input text into words and determining whether or not the words are legal for the language. This scheme imposes language design constraints on the engineer, since words do not have to be delimited to be recognized. The constraint is either to design a requirement for delimitation into the language definition or to ensure that no two adjacent words of the language can ever be combined to make a longer, legal word of the language.

Static Tries In C Arrays

I prefer to store my tries as static data in the same file as the code for the lexical analyzer. This arrangement allows each program to use more than one lexical analyzer, as well as eliminates the need for external files. On the other hand, tries have a widely variable number of elements per trie level, making it imperative to use a dynamically-sized data structure. The method I adopted stores each level of the trie as a uniquely named array with

Listing 2

```
(      L_PAREN
)      R_PAREN
,      COMMA
/      F_SLASH
action ACTION
after  AFTER
and    AND
archive ARCHIVE
attributes ATTRIBUTES
before BEFORE
directory DIRECTORY_T
exec   EXEC
files  FILES
hidden HIDDEN
label  LABEL
modified MODIFIED
name   NAME
not    NOT
or     OR
print  PRINT
readonly READONLY
recurs RECURS
search SEARCH
select SELECT
system SYSTEM
{      L_BRACE
|      BAR
}      R_BRACE
```

Figure 2

- 1) Read a character from the file.
Call step 2 with the pointer to the root of the trie, the character read and a pointer to the beginning of the key word save buffer.
- 2) Search for the input character in the current node of the trie.
If the input character is found in the trie then
 Save the input character in the key word save buffer.
 Attempt to read a character from the file.
 If End of File then
 Return End of File.
 If the character found has a child trie pointer then
 Call Step 2 with the child trie pointer, the character read from the file and a pointer to the next byte in the key word save buffer.
 If the return value from the call to step 2 is NOT FOUND then
 Unread the character read from the file
 Return the token value stored with the input character of this call.
 else
 Return the return value of the recursive call.
else If the character is not found then
 Save a NUL in the key word save buffer.
 Return the value NOT_FOUND.

Listing 1

```
typedef struct key_st {
    char  c;                /* String character.      */
    TKNS  token;            /* Token value.           */
    struct key_st *child;   /* Pointer to sub-trie.   */
} NODE;
```

NODE Structure

MEMORY PROBLEMS SOLVED!

Kandu Tools Help You Break The 640K Barrier!

NEW!

SUPER MALLOC

Up to 352K More Memory!

\$149

- Easy to use, one to one malloc replacement.
- Best fit algorithm gets the most out of conventional memory.
- Use HMA, UMB, Video Memory, and EMS Page Frame.
- Full set of support routines, heapwalk etc.

EMPTY SHELL

Run any program from your program!

\$169

- Shell to any program, leaves less than 5K resident!
- Saves and restores screen - text or graphics!
- Swaps your program to expanded memory, extended memory or disk.
- Easy to use, just a single call.

Xmem

A complete virtual memory system!

\$149

- Handle based malloc replacement.
- Manages expanded, extended, and can swap to disk.
- Eliminates memory fragmentation problems.
- Optionally emulates Windows & Macintosh memory managers.

MEMORY TOOLKIT

Easy access to all memory!

\$99

- Create multi-megabyte arrays in expanded, extended, or on disk.
- Fast - 100% assembly language, dynamic array expansion.
- Read and write expanded or extended memory with simple file like calls.
- Complete set of calls to EMS and XMS managers.

To Order Call:

1-800-755-7973

Visa / MasterCard / C.O.D.

International: (609) 587-7973

Fax: (609) 587-9412

Kandu Inc.

"We Can Do What Others Can't"

16 Cannon Drive Hamilton, New Jersey 08600

ALL OUR PRODUCTS FEATURE: • Source code • Supports Microsoft, Borland & Zortech C/C++
• 30 Day Money-back Guarantee • No Royalties! • 100% Assembler • Free technical support

◆ Request 481 on Reader Service Card ◆

SLATE

Universal
Printer
Driver

Would Text and Graphic Printer support for over 750 Printers (including PostScript) give you a competitive edge?

By including SLATE, you can print both Text and Graphics on over 750 printers (including PostScript, LaserJet III and the new Epson scalable font printers).
Immediately! Painlessly!

You can use SLATE in your product with **no royalties**.

SLATE gets you out of the printer support business. Forever!

Make your product more functional and competitive by using SLATE's advanced text features:

- Support multiple printers on the same system.
- Output to parallel printers, serial printers, DOS files/devices, DOS print spooler, and Novell network printers.
- Include end user's soft fonts.
- Support cartridge fonts.
- Support proportional fonts.
- Support scalable font printers.
- Set exact print positions.
- Color printing
- Kerning, leading, overstrike, underlining, and strike through.
- Automatic character set conversion.

SLATE with Graphics adds advanced graphic printing features:

- Print images from the screen, PCX or TIFF files, or custom image systems.
- Scale and Rotate the printed image.
- Print grey scale and color images.
- Intermix text and graphics.

SLATE is a set of C or Basic libraries with over 170 text printing functions, a Database of over 750 printers, and End User configuration and testing programs.

SLATE with Graphics adds over 60 graphic printing functions.

Call now for a complete developer's information kit. **Order SLATE for only \$299 or SLATE with Graphics for \$448** with our risk free, 30 day return policy.

We accept Visa, MasterCard, COD's or PO's from qualified companies. Source code, maintenance, and site licenses are available.

800-346-3938

**The
Symmetry
Group**

PO Box 26195
Columbus, OH 43226
614-431-2667
FAX 614-431-5734

elements corresponding to the structure in Listing 1.

Storing tries as source code in the lexical analyzer can make for very large source files. Even a small language can have a search trie that is 1100 lines of source code. Such a file compiles to a relatively small amount of data, but just as no program is ever fast enough, no program is ever small enough either. Roughly half to two-thirds of the memory that a trie uses is for storing pointers to sub-tries. To minimize the memory requirements of a search trie

after compilation, you should exploit your compiler's options to group data together. Doing so lets you use a smaller pointer size to reference that data.

In order to make creating and maintaining lexical analyzers that use tries easier, I've written a program that accepts a text file of token words and token define names, creates the trie in memory, and dumps the trie to standard out as static C data arrays. This scheme facilitates adding or deleting a word from the trie.

Listing 3

```
/******
 * Module   :   Lexical Analyzer --- Header file containing token value
 *           :   enumeration, type definitions and function prototypes for
 *           :   the lexical analyzer functions.
 *
 *           :   Copyright (C) 1990 John W. M. Stevens, All Rights Reserved
 *
 * Author    :   John W. M. Stevens
 *****/

#if !defined( LEXICAL_ANALYZER )
#define LEXICAL_ANALYZER 1

#include <dos.h>

#define TRUE 1
#define FALSE 0
#define ERROR -1
#define OK 0

#define PATH_SZ 65
typedef unsigned int UINT;
typedef unsigned char UCHAR;
typedef char PATH[PATH_SZ];

/* Definition of structure filled in and returned by lex. */
typedef struct {
    char str[257];
    long no;
    struct time ftime;
    struct date fdate;
} TOKEN;

/* Token defines. */
enum tkn_en {
    STRING = 128,
    NUMBER,
    TIME,
    DATE,
    L_PAREN,
    R_PAREN,
    COMMA,
    F_SLASH,
    ACTION,
    AFTER,
    AND,
    ARCHIVE,
    ATTRIBUTES,
    BEFORE,
    DIRECTORY_I,
    EXEC,
    FILES,
    HIDDEN,
    LABEL,
    MODIFIED,
    NAME,
    NOT,
    OR,
    PRINT,
    READONLY,
    RECURS,
    SEARCH,
    SELECT,
    SYSTEM,
    L_BRACE,
    BAR,
    R_BRACE
};

typedef enum tkn_en TKNS;

/* Function prototypes. */
extern TKNS Lex(TOKEN *);
extern void OpenPrg(char *);
extern void ParsErr(char *);

#endif
/* End of File */
```


Example Lexical Analyzer

The example lexical analyzer uses a trie that contains the keywords and token values defined in Listing 2. The first word on the line is the keyword, and the second word is the token value enumeration label for that keyword. The trie creation program processes this file. The program output is captured in a file that will be included in the source code file for the lexical analyzer.

Listing 3 contains the token value enumeration, function prototypes, and type definitions necessary to use the lexical analyzer. To increase the readability of the parser source code, I've selected enumeration labels that are as similar as possible to the keywords they represent. Because of name space collision with type and/or define names used by the C compiler, some of the enumeration labels are postfixed with the string _T.

(text continued on page 83)

Listing 4

```
/*-----  
* Module : Lexical Analyzer --- Process the input text file into tokens  
*         that the parser can understand.  
*  
*         Copyright (C) 1990 John W. M. Stevens, All Rights Reserved  
*  
* Routines : Lex - Return the next token from the file.  
*            OpenPrg - Open the source file.  
*            ParsErr - Report a parsing error.  
*  
* Author : John W. M. Stevens  
*-----*/  
  
#include <stdio.h>  
#include <stdlib.h>  
#include <ctype.h>  
#include <string.h>  
  
#include "lex.h"  
  
/* Structure of trie branch. */  
typedef struct key_st {  
    char c;  
    TKNS token;  
    struct key_st *child;  
} NODE;  
  
/* Constants local to this file. */  
#define MAX_STR 256  
#define NOT_FND -2  
  
/* Object Data. */  
static char word[MAX_STR + 1]; /* Last string analyzed. */  
static char PrvWd[MAX_STR + 1]; /* Previous word. */  
static int LnNo = 0; /* The current line number in the file. */  
static FILE *PrgFl; /* File pointer. */
```

Developing GUI Applications for Multiple Platforms has Never Been Easier

Introducing WinTRAN™, the multi-platform GUI development environment that lets you manipulate **named visual objects** from your C or C++ code. As a result, applications developed with WinTRAN contain from *one-half* to *one tenth* the lines of code of GUI applications developed with any other approach. WinTRAN's powerful visual interface builder lets you interactively build your applications' user interface for all popular GUI environments: Windows, PM, OSF/Motif, OpenLook and Macintosh.

It's never been easier to develop GUI applications – for just one platform, or for all.

Call today for more information **1-800-257-4888**



2483 Old Middlefield Way, Suite 224, Mountain View CA 94043

WinTRAN is a trademark of Guideware Corporation. All other trademarks are the property of their respective owners.



Listing 4 — Cont'd

```
/* Trie data structure containing all the keywords and punctuation
marks for
the language being tokenized.
*/
```

```
static
NODE T5[2] = {
    { 'n',          2, NULL },
    { 'n',          ACTION, NULL }
};
```

```
static
NODE T4[2] = {
    { 'o',          2, NULL },
    { 'o',          0, T5 }
};
```

```
static
NODE T3[2] = {
    { 'i',          2, NULL },
    { 'i',          0, T4 }
};
```

```
static
NODE T2[2] = {
    { 't',          2, NULL },
    { 't',          0, T3 }
};
```

```
static
NODE T8[2] = {
    { 'r',          2, NULL },
    { 'r',          AFTER, NULL }
};
```

```
static
NODE T7[2] = {
    { 'e',          2, NULL },
    { 'e',          0, T8 }
};
```

```
static
NODE T6[2] = {
    { 't',          2, NULL },
    { 't',          0, T7 }
};
```

```
static
NODE T9[2] = {
    { 'd',          2, NULL },
    { 'd',          AND, NULL }
};
```

```
static
NODE Te[2] = {
    { 'e',          2, NULL },
    { 'e',          ARCHIVE, NULL }
};
```

```
static
NODE Td[2] = {
    { 'v',          2, NULL },
    { 'v',          0, Te }
};
```

```
static
NODE Tc[2] = {
    { 'i',          2, NULL },
    { 'i',          0, Td }
};
```

```
static
NODE Tb[2] = {
    { 'h',          2, NULL },
    { 'h',          0, Tc }
};
```

```
static
NODE Ta[2] = {
    { 'c',          2, NULL },
    { 'c',          0, Tb }
};
```

```
static
NODE T16[2] = {
    { 's',          2, NULL },
    { 's',          ATTRIBUTES, NULL }
};
```

```
static
NODE T15[2] = {
    { 'e',          2, NULL },
    { 'e',          0, T16 }
};
```

```
static
NODE T14[2] = {
    { 't',          2, NULL },
    { 't',          0, T15 }
};
```

```
static
NODE T13[2] = {
    { 'u',          2, NULL },
    { 'u',          0, T14 }
};
```

```
static
NODE T12[2] = {
    { 'b',          2, NULL },
    { 'b',          0, T13 }
};
```

```
static
NODE T11[2] = {
    { 'i',          2, NULL },
    { 'i',          0, T12 }
};
```



"Without a doubt, OPUS Make
is the hottest make utility
on the market"

Tom Swan, PCWorld

For professional programmers,
OPUS Make is the superior choice
in a make utility. It is the fastest most
full-featured make utility there is.
Features include:

- DOS version uses only 3K memory!
- Multiple directory support.
- Supports: Polytron PVCS™
Burton Systems TLIB™
Microsoft LIB™
SLR Systems OPTLIB™
- Generates automatic response files of
unlimited length for LINK and LIB.
- OPUS MKMF included: An automatic de-
pendency generator that fully under-
stands C preprocessor directives.

Both DOS and OS/2 executables
for only:

\$129 MC, Visa, COD, or
PO. 60 day MBG!

Call 1-800-248-OPUS

International: 415-864-7901

1032 Irving Street, Suite 439, San Francisco, CA 94122

Trademarks owned by their respective companies

Opus Make

Listing 4 — Cont'd

```
static
NODE T10[2] = {
    { 'r', 2, NULL },
    { 'r', 0, T11 }
};

static
NODE Tf[2] = {
    { 't', 2, NULL },
    { 't', 0, T10 }
};

static
NODE T1[6] = {
    { 'c', 6, NULL },
    { 'f', 0, T2 },
    { 'f', 0, T6 },
    { 'n', 0, T9 },
    { 'r', 0, Ta },
    { 't', 0, Tf }
};

static
NODE T1b[2] = {
    { 'e', 2, NULL },
    { 'e', BEFORE, NULL }
};

static
NODE T1a[2] = {
    { 'r', 2, NULL },
    { 'r', 0, T1b }
};

static
NODE T19[2] = {
    { 'o', 2, NULL },
    { 'o', 0, T1a }
};
```

```
static
NODE T18[2] = {
    { 'f', 2, NULL },
    { 'f', 0, T19 }
};

static
NODE T17[2] = {
    { 'e', 2, NULL },
    { 'e', 0, T18 }
};

static
NODE T23[2] = {
    { 'y', 2, NULL },
    { 'y', DIRECTORY_T, NULL }
};

static
NODE T22[2] = {
    { 'r', 2, NULL },
    { 'r', 0, T23 }
};

static
NODE T21[2] = {
    { 'o', 2, NULL },
    { 'o', 0, T22 }
};

static
NODE T20[2] = {
    { 't', 2, NULL },
    { 't', 0, T21 }
};
```

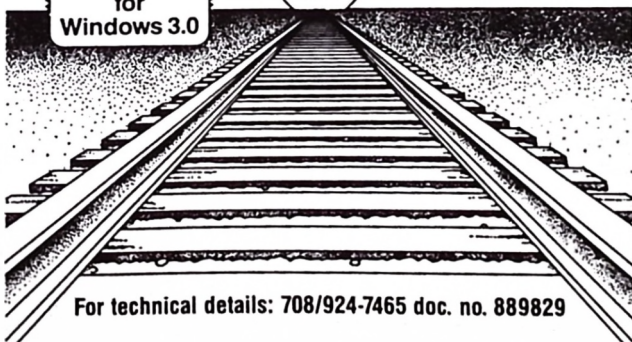
AccSys™ for Paradox.

Because you shouldn't
be stuck
pulling the engine.

- 100% compatibility guaranteed
- Support for ALL popular C & C++ compilers
- Twice the speed and half the size (DBA, May '91)
- C Source available for all memory models

**Copia
International Ltd.**
VOICE: 708/682-8898
FAX: 708/665-9841
Wheaton, Illinois 60187

Ideal
for
Windows 3.0



For technical details: 708/924-7465 doc. no. 889829

PANEL[®] Plus II



Screen Generator and User Interface Library for 'C'

PANEL Plus II features: Interactive screen design editor • C and MS-FORTRAN code generators • library of user-interface functions • pop-up fields/windows • multi-line fields • mouse support • scrolling regions/scroll bars • help boxes • pull-down menus • custom validation • shadow borders • complete library source, with low-level source and headers for all supported systems (DOS, OS/2, Unix, VMS) • interfaces to popular PC graphics libraries • OS/2 DLL • no royalties.

Also available: Utility Source Option which allows PANEL Plus II to be ported to any suitable development platform. *New in v2.1:* CUA/SAA key assignments and edit functions, CUA 'function key area' support, new menu styles, word-wrap editing, QH on-line help, DOS-extender interfaces, extensive example programs.

Panel Plus II license fee per programmer: \$495 (including library source and 700-page manual). **Panel Plus II** with utility source: \$1195. Call for free demo disk. Instant info via fax: dial (708) 924 7465, press 3702#

Roundhill Computer Systems Limited

1342 Avalon Court, Wheaton IL 60187
Telephone: (708) 690 3737 • Fax: (708) 665 9841 • BDX: join roundhill
UK Head Office: POB 14 Marlborough, Wiltshire SN8 1LG • Tel: 0472 84535 • Fax: 0672 84525

Listing 4 — Cont'd

```
static
NODE T1f[2] = {
  { 'f', 2, NULL },
  { 'c', 0, T20 }
};

static
NODE T1e[2] = {
  { 'e', 2, NULL },
  { 'f', 0, T1f }
};

static
NODE T1d[2] = {
  { 'd', 2, NULL },
  { 'r', 0, T1e }
};

static
NODE T1c[2] = {
  { 'c', 2, NULL },
  { 'i', 0, T1d }
};

static
NODE T26[2] = {
  { 'c', 2, NULL },
  { 'c', EXEC, NULL }
};

static
NODE T25[2] = {
  { 'e', 2, NULL },
  { 'e', 0, T26 }
};
```

```
static
NODE T24[2] = {
  { 'x', 2, NULL },
  { 'x', 0, T25 }
};

static
NODE T2a[2] = {
  { 's', 2, NULL },
  { 's', FILES, NULL }
};

static
NODE T29[2] = {
  { 'e', 2, NULL },
  { 'e', 0, T2a }
};

static
NODE T28[2] = {
  { 'l', 2, NULL },
  { 'l', 0, T29 }
};

static
NODE T27[2] = {
  { 'i', 2, NULL },
  { 'i', 0, T28 }
};

static
NODE T2f[2] = {
  { 'n', 2, NULL },
  { 'n', HIDDEN, NULL }
};

static
NODE T2e[2] = {
  { 'e', 2, NULL },
  { 'e', 0, T2f }
};

static
NODE T2d[2] = {
  { 'd', 2, NULL },
  { 'd', 0, T2e }
};

static
NODE T2c[2] = {
  { 'd', 2, NULL },
  { 'd', 0, T2d }
};

static
NODE T2b[2] = {
  { 'i', 2, NULL },
  { 'i', 0, T2c }
};

static
NODE T33[2] = {
  { 'l', 2, NULL },
  { 'l', LABEL, NULL }
};

static
NODE T32[2] = {
  { 'e', 2, NULL },
  { 'e', 0, T33 }
};

static
NODE T31[2] = {
  { 'b', 2, NULL },
  { 'b', 0, T32 }
};
```

Moving from FORTRAN to C?

FOR_C™ delivers
C translations unmatched
for readability and style!

New
Version
3.2

- FOR_C translates to standard C functions like `fprintf()` and `strcpy()`, with less dependence on custom functions. It checks arguments for consistent usage, selecting pass by value translations when possible, and produces excellent C code! Popular FORTRAN extensions like VAX F-77 with structures are supported, also making FOR_C a valuable porting tool.
- We include source to very portable runtime libraries so you can run your C code across different environments!
- Cost effective after only a 1000 line translation, FOR_C makes machine conversion truly practical. Don't just take our word for it — put FOR_C to the test and judge the results!
- Call for information and combined product discounts.
- Ask about **FOR_STRUCT** and **PRO_STRUCT**, our new utilities to convert spaghetti FORTRAN into fully structured code!

COBALT BLUE

875 Old Roswell Rd., Suite D-400
Roswell, GA 30076, USA
TEL (404) 518-1116, FAX (404) 640-1182

◆ Request 105 on Reader Service Card ◆

Listing 4 — Cont'd

```

static
NODE T30[2] = {
    { 'i', 2, NULL },
    { 'a', 0, T31 }
};

static
NODE T3a[2] = {
    { 'd', 2, NULL },
    { 'd', MODIFIED, NULL }
};

static
NODE T39[2] = {
    { 'e', 2, NULL },
    { 'e', 0, T3a }
};

static
NODE T38[2] = {
    { 'i', 2, NULL },
    { 'i', 0, T39 }
};

static
NODE T37[2] = {
    { 'f', 2, NULL },
    { 'f', 0, T38 }
};

static
NODE T36[2] = {
    { 'i', 2, NULL },
    { 'i', 0, T37 }
};

static
NODE T35[2] = {
    { 'd', 2, NULL },
    { 'd', 0, T36 }
};

static
NODE T34[2] = {
    { 'o', 2, NULL },
    { 'o', 0, T35 }
};

static
NODE T3d[2] = {
    { 'e', 2, NULL },
    { 'e', NAME, NULL }
};

static
NODE T3c[2] = {
    { 'm', 2, NULL },
    { 'm', 0, T3d }
};

static
NODE T3e[2] = {
    { 't', 2, NULL },
    { 't', NOT, NULL }
};

static
NODE T3b[3] = {
    { 'a', 3, NULL },
    { 'a', 0, T3c },
    { 'o', 0, T3e }
};

```

```

static
NODE T3f[2] = {
    { 'r', 2, NULL },
    { 'r', OR, NULL }
};

static
NODE T43[2] = {
    { 't', 2, NULL },
    { 't', PRINT, NULL }
};

static
NODE T42[2] = {
    { 'n', 2, NULL },
    { 'n', 0, T43 }
};

static
NODE T41[2] = {
    { 'i', 2, NULL },
    { 'i', 0, T42 }
};

static
NODE T40[2] = {
    { 'r', 2, NULL },
    { 'r', 0, T41 }
};

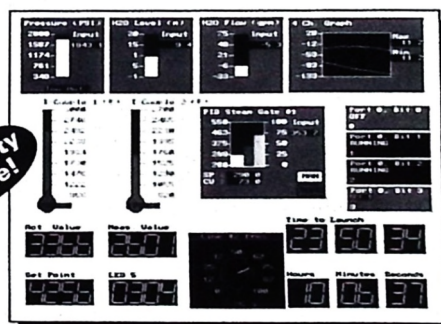
static
NODE T4a[2] = {
    { 'y', 2, NULL },
    { 'y', READONLY, NULL }
};

```

GIL 2.0 Only \$249

Graphical Instrument Library

The universal library for building
fast graphical displays for data
acquisition and control.



Royalty Free!

Rich set of instruments includes:
Dial gauges, bar gauges, thermometers, seven-segment
displays, strip-charts, annunciators, alarms, signal
conditioning, timing, PID control and more!

All instruments are fully scalable. Virtual coordinates
allow programs to run unmodified on over 27 video modes!
Works with any data acquisition hardware,
even serial-based systems!

One version supports Turbo C, Microsoft C, QuickC,
QuickBASIC, Turbo Pascal and others available soon.

For a free demo, call

(404) 352-4788

**Advanced
Design
Solutions**

1920 Moores Mill Rd., Atlanta, GA 30318

◆ Request 142 on Reader Service Card ◆

Listing 4 — Cont'd

```
static
NODE T49[2] = {
    { ' ', 2, NULL },
    { 'l', 0, T4a }
};

static
NODE T48[2] = {
    { ' ', 2, NULL },
    { 'n', 0, T49 }
};

static
NODE T47[2] = {
    { ' ', 2, NULL },
    { 'o', 0, T48 }
};

static
NODE T46[2] = {
    { ' ', 2, NULL },
    { 'd', 0, T47 }
};

static
NODE T4d[2] = {
    { ' ', 2, NULL },
    { 's', RECURS, NULL }
};

static
NODE T4c[2] = {
    { ' ', 2, NULL },
    { 'r', 0, T4d }
};
```

```
static
NODE T4b[2] = {
    { ' ', 2, NULL },
    { 'u', 0, T4c }
};

static
NODE T45[3] = {
    { ' ', 3, NULL },
    { 'a', 0, T46 },
    { 'c', 0, T4b }
};

static
NODE T44[2] = {
    { ' ', 2, NULL },
    { 'e', 0, T45 }
};

static
NODE T52[2] = {
    { ' ', 2, NULL },
    { 'h', SEARCH, NULL }
};

static
NODE T51[2] = {
    { ' ', 2, NULL },
    { 'c', 0, T52 }
};

static
NODE T50[2] = {
    { ' ', 2, NULL },
    { 'r', 0, T51 }
};

static
NODE T55[2] = {
    { ' ', 2, NULL },
    { 't', SELECT, NULL }
};

static
NODE T54[2] = {
    { ' ', 2, NULL },
    { 'c', 0, T55 }
};

static
NODE T53[2] = {
    { ' ', 2, NULL },
    { 'e', 0, T54 }
};

static
NODE T4f[3] = {
    { ' ', 3, NULL },
    { 'a', 0, T50 },
    { 'l', 0, T53 }
};

static
NODE T59[2] = {
    { ' ', 2, NULL },
    { 'm', SYSTEM, NULL }
};

static
NODE T58[2] = {
    { ' ', 2, NULL },
    { 'e', 0, T59 }
};
```

BAS_C

BASIC to C translator

- inputs BASICA, Quick BASIC
- outputs MS/Quick/Turbo C[++]
- CUTOMIZER™ for other BASIC
- restructures any spaghetti code
- indented, scoped, modularized C
- syntax-error free, portable C
- runtime library written in C
- run on MSDOS, XENIX, UNIX
- 1,000 copies sold since 1986
- FREE DEMO DISK, INFO

Gotoless Conversion

7105 Dee Cole Drive, The Colony, TX 75056

Phone (214) 625 - 2323

Fax: 214-370-2612 BBS: 214-625-6905

Listing 4 — Cont'd

```
static
NODE T57[2] = {
    { ' ', 2, NULL },
    { 't', 0, T58 }
};

static
NODE T56[2] = {
    { ' ', 2, NULL },
    { 's', 0, T57 }
};

static
NODE T4e[3] = {
    { ' ', 3, NULL },
    { 'e', 0, T4f },
    { 'y', 0, T56 }
};

static
NODE T0[21] = {
    { ' ', 21, NULL },
    { '(', L_PAREN, NULL },
    { ')', R_PAREN, NULL },
    { ',', COMMA, NULL },
    { '/', F_SLASH, NULL },
    { 'a', 0, T1 },
    { 'b', 0, T17 },
    { 'd', 0, T1c },
    { 'e', 0, T24 },
    { 'f', 0, T27 },
    { 'h', 0, T2b },
```

```
{ 'l', 0, T30 },
{ 'm', 0, T34 },
{ 'n', 0, T3b },
{ 'o', 0, T3f },
{ 'p', 0, T40 },
{ 'r', 0, T44 },
{ 's', 0, T4e },
{ '{', L_BRACE, NULL },
{ '|', BAR, NULL },
{ '}', R_BRACE, NULL }
};
```

```
/*-----
Routine : OpenPrg() --- Open the ASCII text file
         that contains the back up program.

Inputs  : FileNm - File name of source file.
-----*/
```

```
void OpenPrg(char *FileNm)
{
    /* Open the program script file. */
    if ((PrgFl = fopen(FileNm, "rt")) == NULL)
    {
        fprintf(stderr, "OpenPrg (fopen) : Could not open
file '%s' for "
            "reading.\n", FileNm);
        exit(-1);
    }

    /* Initialize object variables. */
    *word = *PrvWd = '\0';
}
```

End the DOS Nightmare! Wake Up to *InCommand*™

If working with DOS during the day is keeping you up at night, it's time to get *InCommand*!

PCM Magazine Publisher's Pick 10/91

"A powerful supplement to DOS.... Microsoft should have thought of these InCommand utilities."

- **Text Search:** full screen browse, line modes, whole-word-only option, more features than Norton
- **Execute:** run your own programs and batch files as if they had all the InCommand file selection capabilities
- **Move** (without copying), **Rename**, **Delete**, **Directory** (sorted, including file find): files, directories, or entire trees
- **Copy:** multiple floppies in one command; incremental backups Up to 40% faster to floppies than XCOPY
- On-line InCommand & DOS reference ("Better than DOS 5.0's")
- Physical directory sort, network compatible, and much more!

Productivity You Only Dreamed Of, Until Now!

Run right from the DOS prompt. No menus to slow you down.

Multiple *s in file and directory wildcards.

Select (and exclude) multiple wildcard patterns at once.

Select files BEFORE, AFTER, or ON any date/time, BIGGER or SMALLER than any size, with or without any attributes.

Process entire directory trees as easily as one file.

*Quoted from PCM Magazine, 2/92.

**Much more! Only \$60
Call now for demo disk.**

You can rest easy with *InCommand*.

Inductive Logic



P.O. Box 26238
San Diego, CA 92196
(619) 578-5146

Intelligent Software for Every User

◆ Request 137 on Reader Service Card ◆

TIFF, PCX, & GIF SDK FOR WINDOWS 3.0

**Add support to your Windows Application
without learning file formats!**

TIFF SDK For Windows is a complete set of easy-to-use tools for adding TIFF 5.0 support to your application without learning the complexity of Tag Image File Format. Features include support for: Group 3 and Group 4 fax file formats, LZW and Packbit compression, 24-bit color images, chained images and device independent bitmaps. Includes full documentation plus a sample application complete with source code. **Only \$299.95**

PCX & GIF SDK For Windows adds the latest 24-bit color support to your application. Includes documentation plus a sample program complete with source code. **Each Only \$149.95**

IMAGE SDK For Windows contains functions to read, write, and manipulate Microsoft Windows CLP, WMF, BMP, and MSP file formats. Zooms, inverts, & rotates bitmaps. Comes with documentation, sample program and source code. **Only \$199.95**

NEW!! Color Generic Device Driver for Windows Now you can output high resolution TIFF, PCX & GIF, 100-600 DPI, compatible with any Windows application with capability to print. **Only \$199.95**

All SDK's are compatible with Visual Basic, SQL
Windows, Actor, MS Excel and Borland C++



BLACK ICE SOFTWARE INC.

Crane Road, Somers N.Y. 10589

TEL: (914) 277-7006 FAX: (914) 276-8418

VISA/M.C./C.O.D/CHECK

◆ Request 300 on Reader Service Card ◆

Code Generator for Rule-Based Systems

File	Rules	Database	Export	Rules	Verify	Print	Quit
stage1.lisp	short	high	high	high	high	high	high
stage2.lisp	short	high	high	high	high	high	high
stage3.lisp	short	high	high	high	high	high	high
stage4.lisp	short	high	high	high	high	high	high
stage5.lisp	short	high	high	high	high	high	high
stage6.lisp	short	high	high	high	high	high	high
stage7.lisp	short	high	high	high	high	high	high
stage8.lisp	short	high	high	high	high	high	high
stage9.lisp	short	high	high	high	high	high	high
stage10.lisp	short	high	high	high	high	high	high
stage11.lisp	short	high	high	high	high	high	high
stage12.lisp	short	high	high	high	high	high	high
stage13.lisp	short	high	high	high	high	high	high
stage14.lisp	short	high	high	high	high	high	high
stage15.lisp	short	high	high	high	high	high	high
stage16.lisp	short	high	high	high	high	high	high
stage17.lisp	short	high	high	high	high	high	high
stage18.lisp	short	high	high	high	high	high	high
stage19.lisp	short	high	high	high	high	high	high
stage20.lisp	short	high	high	high	high	high	high

Build, test, verify, document, generate C source code, and embed rule-based systems *quickly* with...

Knowledge Shop™

Rules are used in industry every day. If you haven't tried rule-based systems in your programs, consider where others have: *policy and loan evaluation, stock and bond selection, equipment maintenance and diagnostics, systems design, part design, process monitoring and control, product selection and configuration, training* - the list goes on and on.

Knowledge Shop brings fast, portable rule-based reasoning to any C or C++ program. With **Knowledge Shop's** simple visual approach, you link nodes into a dataflow diagram that models the problem, import or create rules to specify processing at each node, then automatically generate portable, compact ANSI C source code that seamlessly embeds into your program.

"I don't know where I'd be without **Knowledge Shop**" Bruce Harper, **NextWave Software**

- Import or merge text rules at any time
- Automatic color-coded rule debugging
- Automatic generation of missing rules to cover gaps
- Automatic generation of documentation
- Automatic generation of ANSI C source code
- Complete portability of generated C code
- Seamless embedding into any C or C++ program
- Hooks for input, output, or error processing

Honda of America •
NextWave Software • Los Alamos National
Lab • US Dept. of Commerce • Microway
Software • Marketing Resources • Lehigh
University • Univ. of Connecticut...

Try **Knowledge Shop** for 15 days for \$30!
Includes the full program and manual. Keep it and
pay the \$465 balance or return it with no further
obligation

30-Day Money Back Guarantee!



\$495

No
Royalties!

1-203-632-7570

Fax: 1-203-635-3839

Decision System Software
160 West Street
Cromwell, Connecticut 06416



Requires IBM compatible, 640K, CGA, or better color display, DOS
2.1 or higher. **Knowledge Shop** is a trademark of Decision System
Software. CT residents add 6% sales tax

♦ Request 112 on Reader Service Card ♦

Listing 4 — Cont'd

```

/*-----
Routine : ParseErr() --- Report a parse error.
Inputs  : Err - Error string.
*/-----

void ParseErr(char *Err)
{
    /* Print line number and error message. */
    fprintf(stderr, "Error in Line: %d, %s.\n", LnNo + 1, Err);

    /* If there is a previous word, show it. */
    if (*word)
        fprintf(stderr, "\t0n or after word '%s'\n", word);
    exit(-1);
}

/*-----
Routine : TrieSrch() --- Search the trie for a key word.
Inputs  : Trie - The trie level pointer.
          ch   - The current character to search for.
          WordPtr - The pointer to the current byte of the word buffer.
Returns : Returns either a token value or
          NOT_FND - For key word not found.
          EOF     - For end of file.
*/-----

static
int TrieSrch(NODE *Trie,
             int ch,
             char *WordPtr)
{
    register int mid; /* Mid point of array piece. */
    register TKNS ret; /* Return value of comparison. */

    auto int lo; /* Limits of current array piece for */
    auto int hi; /* binary search. */

    /* Make sure that input is lower case. */
    ch = tolower(ch);

    /* Search for a token. */
    hi = Trie[0].token - 1;
    lo = 1;
    do
    {
        /* Find mid point of current array piece. */
        mid = (lo + hi) >> 1;

        /* Do character comparison. */
        ret = ch - Trie[mid].c;

        /* Fix the array limits. */
        if (ret <= 0)
            hi = mid - 1;
        if (ret >= 0)
            lo = mid + 1;
    } while (hi >= lo);

    /* If the character matches one of the entries in this level and this
     * entry has a child, recurse. If a match is found but the matching
     * entry has no child, return the token value associated with the
     * match. If the return value from the recursive call indicates that
     * no match was found at a lower level, return the token value
     * associated with the match at this level of the trie.
     */
    if (ret == 0)
    {
        /* Save the current character in the buffer for error reporting. */
        *WordPtr++ = ch;
    }
}

```

The C Users Journal

As the leading international resource for C programmers, *The C Users Journal* provides you with practical tools and useful information. Every four weeks *The Journal* delivers "how to" insight, written by programmers for programmers.

As the successor to *The CUG Newsletter* (1981) and *The C Journal* (1984), *The C Users Journal* has published continuously since 1987.

Subscribe now — only \$29.95 for a full year of *The C Users Journal*!

Fill out the handy order form (on reverse side), or use one of our postage paid cards (found elsewhere in *The Journal*).

The C Users' Group

The C Users' Group collects and maintains public domain source code (our library now includes over 200 volumes), translates and distributes materials from the library in more than 100 different formats, and helps members find the programs, consultants and other resources they need to do their job well.

The CUG User Supported C Source Code Directory provides clear, file-by-file descriptions of volumes in the CUG library. Each volume of *The Directory* includes an index with keywords and extensive articles describing the most important volumes.

Use Our Handy Order Form on Reverse Side

The C Users Bookstore

The C Users Bookstore is your resource center for C programming books. We offer:

- One stop shopping
- Hard-to-find books
- One low price (no hidden fees!)

Don't waste time wandering from bookstore to bookstore! The C Users Bookstore maintains a list of the best books on C and Unix programming, books like *Illustrated C*, *MS-DOS System Programming, Second Edition*, and *C++ Programming Guidelines*.

The C Users Bookstore makes it easy — just use our order form (opposite page) to pick the books you want. Drop your order in the mail and relax. We mail most orders within 48 hours of receipt.

Use Our Handy Order Form On Reverse Side

1601 West 23rd St., Suite 200, P.O. Box 3127,
Lawrence, KS 66046-0127 (913) 841-1631

The C Users Order Form

	Quantity	Code #	Item	Unit Price	Total
C Users Journal			1-year subscription (12 issues)	\$29.95 US, \$54 CAN/MEX, \$65 overseas	
			2-year subscription (24 issues)	\$56 US, \$89 CAN/MEX, \$123 overseas	
			3-year subscription (36 issues)	\$79 US, \$122 CAN/MEX, \$175 overseas	
			Magazine Code Disk (Mo: Yr:)	\$5.00	
			Magazine Code Disks — 1991 (V.9)	\$20.00	
			Magazine Code Disks — 1990 (V.8)	\$20.00	
			CUJ Back Issue (Mo: Yr:)	\$7.50, \$11.00 overseas	

	Quantity	Code #	Title	Unit Price	Total
Books See below for overseas shipping charges.		CAT1	The CUG Directory - Vol. I	\$10.00	
		CAT2	The CUG Directory - Vol. II	\$10.00	
		CAT18	CUG Directory Volumes I & II	\$18.00	

	Quantity	Code #	Disk Format/Size (MS-DOS, etc.)	# disks per volume	Unit Price	Total
Disks See below for overseas shipping charges. NOTE: Please indicate format and size of disks ordered.				() x \$4.00		
				() x \$4.00		
				() x \$4.00		
				() x \$4.00		

Please send more information about:

- ☐ SysAdmin
☐ The C Users Journal ☐ Windows™/DOS
☐ The C Users Bookstore ☐ The C Users' Group

All payments must be in US dollars (Mastercard/VISA accepted)
NOTE: Call 913-841-1631 for special shipping!

Name _____
 Company _____
 Address _____

 City _____
 State _____ Zip Code _____
 Country _____
 Daytime Phone _____

☐ I am a current subscriber

OVERSEAS SHIPPING CHARGES	BOOKS - Add 45% + \$3.50 s&h	
	DISKS - Add 30% + \$3.50 s&h	
shipping charge in North America (\$3.50)		

TOTAL

☐ MC ☐ Visa Expiration Date _____
 Card # _____
 Signature _____

Mail to: The C Users Journal, 1601 W. 23rd St., Suite 200
P.O. Box 3127, Lawrence KS 66046-0127

— For fastest service, call (913) 841-1631 —

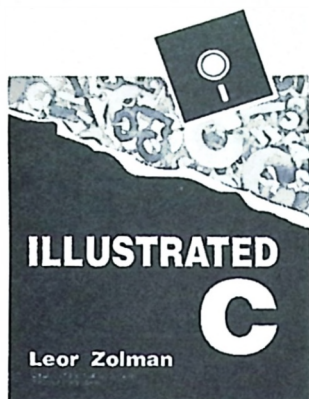


Users Bookstore

The Resource Center for C Programming Books

Hard-to-find titles and titles you won't find anywhere else

C++ • UNIX • ANSI/Standard C • Turbo C • Windows • X Window



Covers UNIX and DOS platforms

Illustrated C

By Leor Zolman

Illustrated C explores the construction of several different applications, from start to finish. Through a focus on building useful, practical programs, this book gives the C programmer the "why and how" of application design and development. Each program is exhaustively annotated in a clear, readable style.

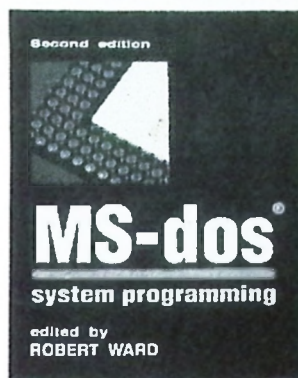
Illustrated C is a tutorial for the novice to intermediate programmer and a toolbox for all C programmers.

R&D Publications, 1992, 320 pp.

ISBN 0-923667-21-0

W34\$29.95

W36 w/ disk\$39.95



MS-DOS System Programming / 2nd edition

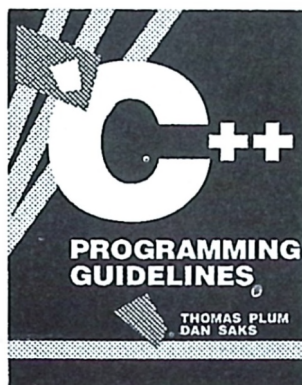
Edited by Robert Ward

This highly technical, focused work is written by working programmers for working programmers and designed to save the reader hours of work. How-to topics include: critical error handling, customizing the DOS boot strap, interrupt-driven I/O, manipulating environmental variables, event timing, writing TSRs and device drivers, and much more. The final chapter is an annotated bibliography of books designed to help the serious programmer develop a personal library of professional MS-DOS literature.

R&D Publications, 1991, 240 pp.

ISBN 0-923667-20-2

Z82\$24.95



C++ Programming Guidelines

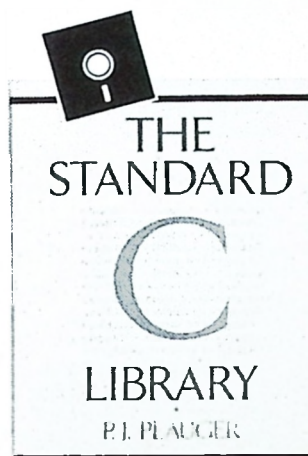
By Thomas Plum and Dan Saks

C++ Programming Guidelines gives professional C++ developers useful information for writing portable and maintainable C++ programs. Written for the professional C and C++ programmer, this book aids the production of efficient, portable programs which can be understood and maintained by other developers. Users learn both lexical layout and more fundamental semantics for data and variables, operators, control statements, and functions. C++ Programming Guidelines is arranged in a "manual-page" format for easy reference.

Plum-Hall, 1992, 265 pp.

ISBN 0-911537-10-4

W33\$34.95



The Standard C Library

By P.J. Plauger

The Standard C Library shows you how to use all of the library functions mandated by the ANSI and ISO Standards for the programming language C. To help you understand how to use the library, this book also shows you how to implement it. Approximately 9,000 lines of tested, working code that is highly portable across diverse computer architectures is included.

Prentice Hall, 1991, 498 pp.

ISBN 0-13-131509-9

Z81\$28.00

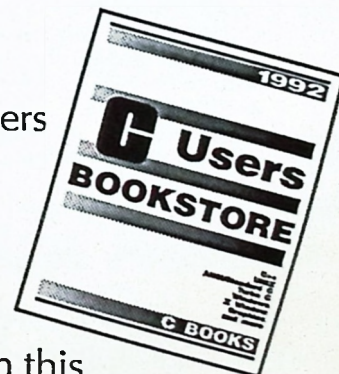
DK1 w/ disk\$77.95

Use the order form on the opposite page to order these titles or any of 150 books in the Bookstore.

FREE 1992 C Users Bookstore Guide

Get your complete listing and description of all the books in the C Users Bookstore. The 24-page reference guide is yours for the asking. Titles are arranged by subject for easy reference and are indexed by title and author.

CALL TODAY — (913) 841-1631. Or, use the Reader Service card in this magazine. Request 153 on the Reader Service Card.



The C Users' Group

The C Users' Group collects and maintains the public domain C source code and shareware listed below. Distribution fee is \$4.00 per disk. There is a \$3.50 shipping and handling charge per order U.S.; 30% overseas. If there are multiple disks in a volume, that number is indicated in parentheses. Due to updates, the number of disks per volume is subject to change without notice. The CUG Library supports code written for MS-DOS, CP/M and UNIX. Volumes are continually added.

For volumes 101-156 see the CUG Library Directory.

- | | | | |
|---|--|--|---|
| 157 - QE for MSDOS | 209 - Simplex Curve Fitting | 259 - Console I/O & Withers Tools | 310 - Little Smalltalk for MS-DOS (2) |
| 158 - QE for BDS C | 210 - Simulation and Loadable BIOS for CP/M | 260 - Zmodem, CU, tty library | 311 - DB package |
| 159 - Adventure, MSDOS | 211 - Search, Sort and Merge | 261 - 68k Cross Assembler MS-DOS | 312 - Make-Maker |
| 160 - Programs from Learning to Program in C (Plum Hall) | 212 - Simulation for BDS | 262 - Ramey Tools | 313 - STEVIE |
| 161 - Programs from Efficient C (Plum Hall) | 213 - Van Nuys Enhancements | 263 - C wndw Toolkit (2) | 314 - MNP C Library |
| 162 - Mchlp 80 | 214 - File Display Util | 264 - NRO & Other PC Tools | 315 - FTGRAPH |
| 163 - Small C for MSDOS (PCC, Toronto) (2) | 215 - BBS for BDS C | 265 - cpio Installation Kit + Call CUG for ordering Instructions (2) | 316 - AS8 Cross Assembler |
| 164 - Windows | 216 - Zmodem & Saveram (2) | 266 - Micro PLOX | 317 - Group3 Image Processing |
| 165 - Programs from Reliable Data Structures in C (Plum Hall) | 217 - Spell & Dict Part I | 267 - 8085, 2650 & 2656 Cross Assemblers | 318 - RED (2) |
| 166 - CUG Directory (#101-#155) | 218 - Dict Part II (2) | 268 - Unicorn Library Turbo C (2) | 319 - CPP (2) |
| 167 - Windows, Unix-like Util | 219 - 6502 Cross Assemblers | 269 - Unicorn Library MSC | 320 - Convolution Image Process |
| 168 - Simple Database (PC-SIG 147) | 220 - Windows BOSS (3) | 270 - Miscellany X | 321 - Mouse Trap Library |
| 169 - Miscellany IV (PC-SIG 314) | 221 - 6809 C for flex | 271 - Steven's Library Turbo C (2) | 322 - Doctor's Tools |
| 170 - Miscellany V (PC-SIG 315) | 222 - Small C for CP/M Doc & Exec | 272 - Steven's Library Datalight C | 323 - Fireworks and Adventure |
| 171 - Miscellany VI (PC-SIG 341) | 223 - Small C for CP/M Source (2) | 273 - Turbo C Utilities | 324 - WGCONIO |
| 172 - LEX Part 1 | 224 - Utilix | 274 - Arrays for C | 325 - VGA Graphics Library (2) |
| 173 - LEX Part 2 | 225 - Utilities and Zmath | 275 - Linear Cellular Automata II (2) | 326 - SoftC Database Library (2) |
| 176 - Xlisp v 1.6 (2) | 226 - ART-CEE | 276 - Z80 and 6804 cross assemblers | 327 - Panels for C |
| 177 - Dr. Shaw's DOS Shell | 227 - Portable Graphics (2) | 277 - HP Plotter Library and XMODEM | 328 - WTWG (2) |
| 178 - TVX Source | 228 - Miscellany IX | 278 - CXL Function Library (3) | 329 - UNIX Tools for PC (2) |
| 179 - TVX Doc & Exec. | 229 - Little Smalltalk -UNIX Part 1 | 279 - CROBOTS | 330 - CTask (3) |
| 181 - CFORUM & Lzw | 230 - Little Smalltalk -UNIX Part 2 | 280 - Software Tools Souce | 331 - SE Editor |
| 182 - UNIX BBS | 231 - Little Smalltalk -Unbundled Part 1 | 281 - Unicorn Library v5.1 (2) | 332 - PCurses (2) |
| 183 - PC Tools II | 232 - Little Smalltalk -Unbundled Part 2 | 282 - Quip and Graphics | 333 - gAWK (2) |
| 184 - RunAMD | 233 - Make And Other Utilities | 283 - FAFNIR (3) | 334 - GNUPLOT (3) |
| 185 - Sorts | 234 - XDIR File and Maintenance Utilities | 284 - Portable 8080 Emulator | 335 - Frankenstein Cross Assemblers (4) |
| 186 - Make & Aim | 235 - Overview File and Maintenance Utilities (2) | 285 - Bison for MS-DOS (3) | 336 - EGAPAL/EDIPAL |
| 187 - Functions IV | 236 - Highly Portable Utilities (CUG Starter Disk) | 286 - GRAD for MSC (2) | 337 - Designing Screen Interfaces in C |
| 188 - Utilities VI | 237 - GRAD Graphics - main library | 287 - GRAD for Turbo C (3) | 338 - 68K C Compiler and Assembler (2) |
| 189 - PC Tools III | 238 - GRAD Graphics - demos | 288 - Traveling Salesman, SD and Master Environment Access | 339 - CTRLCLIB (2) |
| 190 - Steve Passe's 68K Assembler | 239 - P.C. Gammon For MSC4 | 289 - Othello | 340 - C-Window |
| 191 - Miscellany VII | 240 - P.C. Gammon For Turbo | 290 - FLEX (2) | 341 - Orbit Propagation |
| 192 - Miscellany VIII | 241 - Inference Engine & Rule Based Compiler | 291 - JJB for Quick C & Turbo C | 342 - I8255 Interface Library |
| 193 - Crypto Toolbox | 242 - Still More Cross Assemblers | 292 - ASxxx C Cross Assemblers (4) | 343 - C Image Processing System (4) |
| 194 - JUGPDS 17 | 243 - DECUS C Preprocessor | 293 - 3D Med. Imaging, Source (2) | 344 - Grab Bag #1 (2) |
| 195 - JUGPDS 18 | 244 - deBruijn | 294 - 3D Med. Imaging, sample images (4) | 345 - TLC/TLF |
| 196 - JUGPDS 19 | 245 - Linear Cellular Automata | 296 - C to C++ Migrator | 346 - ASxxx Cross Assembler Part 2 (2) |
| 197 - MicroEmacs v 3.9 Exec. & Doc. | 246 - Cycles, Mandelbrot Graphics (2) | 297 - Small Prolog | 347 - TAVL Tree |
| 198 - MicroEmacs v 3.9 Source (2) | 247 - Miracl (3) | 298 - PC Curses | 348 - 8048 Disassembler/Z80 Assembler |
| 199 - GED MSDOS text editor | 248 - Micro Spell | 299 - MEL and BP | 349 - Simulation Subroutine Set |
| 200 - Small c Interpreter | 249 - C Spot Run Library (2) | 300 - MAT_LIB | 350 - PCX Graphics Library |
| 201 - MSDOS System Support | 250 - 68K FP Library & Mandelbrot Graphics | 301 - BGI Applications (2) | 351 - UltraWin |
| 202 - KAREL for MSDOS | 251 - Pull Down Menus & Adventure System | 302 - 3-D Transforms | 352 - String and Vlist |
| 203 - YAM for MSDOS | 252 - C Tutor Doc | 303 - MC68K Disassembler | 353 - C++ Tutor (2) |
| 204 - 68000 C Compiler | 253 - C Tutor Source | 304 - ROFF5 | 354 - CES Mouse Tools Library w/ Joystick |
| 205 - Utilities VII | 254 - EGA Graphics Library | 305 - HGA Mandelbrot Explorer /Card Games (2) | 355 - Sherlock for MS-DOS (3) |
| 206 - Checkbook Register Doc & Exec | 255 - DeSmet Carry Routines (2) | 306 - Threads and Synapsys | 356 - Sherlock for Macintosh (3) |
| 207 - Checkbook Register Doc & Source | 256 - C Tutor doc (Turbo C) | 307 - ADU & COMX (Device Driver) | 357 - CSTAR (2) |
| 208 - e for CP/M 68K | 257 - C Tutor Source (Turbo C) | 308 - MSU, REMZ & LIST (2) | 358 - cbase |
| | | 309 - 6809 C Compiler for MS-DOS | 359 - GNU C/C++ for 386 (12) |
| | | | 360 - Uspell (2) |
| | | | 361 - Gadgets and Term |

Be sure to specify format when ordering disk. We can supply these volumes in almost any MS-DOS format (3 $\frac{1}{2}$ " or 5 $\frac{1}{4}$ "), UNIX tar or cpio archives (5 $\frac{1}{4}$ " only), or Macintosh.

The C Users' Group, 1601 W. 23rd St., Suite 200, P.O. Box 3127, Lawrence, KS 66046-0127
(913) 841-1631 • FAX (913) 841-2624

(text continued from page 71)

Listing 4 contains the lexical analyzer. I've extracted this code from a program that acts as a user-programmable file selection shell. The function *OpenPrg()* initializes the lexical analyzer by opening the file that contains the source code to be analyzed. The parser then calls the function *Lex()* repeatedly to get tokens. Each time *Lex()* is called, it begins by reading and throwing away both white space characters (space, tab and newline characters) and comments. When the first character of a suspected keyword is found, it breaks out of the loop and attempts to get either a string constant, integer constant, time, or date.

If the input is not a constant of some type, the trie search function is called. The function *TrieSrch()* begins by attempting to find the input character in the trie node. *TrieSrch()* accepts a pointer to a node of a trie, a character to search for, and a pointer to a buffer for storing the word read from the input file. The function uses a binary search because the characters in a trie node are stored in sorted order.

If the input character is found in the trie node, *TrieSrch()* saves it in the word buffer. If the matching character in the trie node has a pointer to a child trie node, *TrieSrch()* reads another character from the file and calls itself recursively. If the return value from the recursive call indicates that the character was not found, *TrieSrch()* assumes that the input character for this call was the last character of a legal word and unread the character read for the recursive call. The token value of the input character to this call is returned.

If the matching character does not have a pointer to a child trie node, the keyword buffer is NUL-terminated and the token value stored with the matching character is returned. If the input character is not found in the trie node, *TrieSrch()* NUL-terminates the keyword buffer and returns a value indicating that the character was not found.

Figure 2 presents an algorithm in structured English for separating words from an input character stream and searching for them in a search trie.

(text continued on page 85)

Listing 4 — Cont'd

```
/* Are we looking for more characters in this string? */
if ( Trie[mid].child )
{
    /* Get the next character. */
    if ((ch = fgetc( PrgFl )) == EOF)
        return( EOF );

    /* Search next level. */
    if ((ret = TrieSrch(Trie[mid].child, ch, WordPtr)) == NOT_FND)
    {
        ungetc(ch, PrgFl);
        return( Trie[mid].token );
    }
    return( ret );
}
else
```



PC-lint 5.0 presents C Bug # 647

```
void f( n, m )
{
    long nm;

    nm = n * m;
    :
}
```

This program fragment ported from Unix to MS-DOS doesn't work quite right. If you need help figuring out why, give us a call. Refer to Bug # 647.

PC-lint will catch this and many other C bugs. Unlike your compiler, PC-lint looks across all modules of your application for bugs and inconsistencies.

New — Optional Strong Type Checking and variables possibly not initialized.

More than 330 error messages. More than 105 options for complete customization. Suppress error messages, locally or globally, by symbol name, by message number, by filename, etc. Check for portability problems. Alter size of scalars. Adjust format of error messages. Automatically generate ANSI prototypes for your K&R functions.

Attn: Power users with huge programs.

PC-lint 386 uses DOS Extender Technology to access the full storage and flat model speed of your 386. Now fully compatible with Windows 3.0 and DOS 5.0

PC-lint 386 — \$239

PC-lint DOS - OS/2 — \$139

Mainframe & Mini Programmers FlexeLint in obfuscated source form, is available for Unix, OS-9, VAX/VMS, QNX, IBM VM/IMVS, etc. Requires only K&R C to compile but supports ANSI. Call for pricing.

Gimpel Software

3207 Hogarth Lane, Collegeville, PA 19426

CALL TODAY (215) 584-4261 Or FAX (215) 584-4266

30 Day Money-back Guarantee.

PA add 6% sales tax.

PC-lint and FlexeLint are trademarks of Gimpel Software

TauMetric Redefines Productivity with OREGON C++

Productivity in the Small

*Compile directly to object code,
eliminating all intermediate steps.*

- Fast Compilation
- Fast Debugging
- Fast Code

Productivity in the Large

Support for popular dialects.

- K&R C
- ANSI C
- C++ 2.1

Support for popular platforms.

- Sun-4 (SPARC)
- DECstation
- VAX/VMS

Productivity in the long term

- TauMetric Support
- TauMetric Expertise

*You probably have used our
products under other names,
but now you can get our
support directly.*

*TauMetric has developed OEM
C++ products since 1987,
including Oregon C++. We
bought the Oregon Software
product line and will support it
while releasing our exciting
new C++ products.*



**TAUMETRIC
CORPORATION**

8765 Fletcher Parkway, Suite 301
La Mesa, California 91942
(619)697-7607 FAX(619)697-1140
800-874-8501

Listing 4 — Cont'd

```
{
    /* Properly NUL terminate the buffer that the keyword is
     * being saved in and return the token value.
     */
    *WordPtr = '\0';
    return( Trie[mid].token );
}

/* Terminate string in keyword buffer and return not found. */
*WordPtr = '\0';
return( NOT_FND );
}

/*-----
Routine   :  GetNo --- Get a number from the file.
Inputs    :  word   - Pointer to word buffer for error reporting.
Outputs    :  RetNo  - Returns the number read from the file.
Returns    :  Returns the last character read from the file or EOF.
-----*/

static
int  GetNo(char  **word,
           long  *RetNo)
{
    auto      int  c;

    /* Get number. */
    *RetNo = 0L;
    while ((c = fgetc( PrgFl )) >= '0' && c <= '9')
    {
        /* Save character in word buffer. */
        *(*word)++ = c;

        /* Calculate value of number. */
        *RetNo = *RetNo * 10L + (long) (c - '0');
    }
    return( c );
}

/*-----
Routine   :  Lex() --- Get the next key word from the input file.
Outputs    :  sym - The symbolic data read from the file.
Returns    :  Returns the token read or EOF.
-----*/

TKNS  Lex(TOKEN  *sym)
{
    register  int  i;
    register  int  tkn;
    auto      int  ch;
    auto      char  *bf;

    /* Strip comments and white space. If the character read is a '#',
     * every thing to the end of the line is a comment.
     */
    ch = fgetc( PrgFl );
    while (ch == ' ' || ch == '\t' || ch == '\n' || ch == '#')
    {
        /* Process the special characters '#' and '\n'. */
        if (ch == '\n')
            /* End of line, increment the line number. */
            LnNo++;
        else if (ch == '#')
        {
            /* Found a comment character, strip all characters to end
             * of line and increment the line number.
             */
            while (fgetc( PrgFl ) != '\n')
                ;
            LnNo++;
        }
    }
}
```


(text continued from page 83)

Summary

A trie is probably not the most efficient data structure for determining the legality of an input word. A sorted table of strings searched with a binary search would probably be faster and more memory efficient.

So why use a trie if it isn't as fast or efficient as other methods? Since the hardest part of writing a lexical analyzer is in breaking an undifferentiated stream of input characters into words, the beauty of a trie is that it groups characters into words and determines their legality at the same time. It is also, in my opinion, a more elegant solution. This alone is reason enough for me to use a trie. □

Listing 4 — Cont'd

```

/* Get the next character. */
ch = fgetc( PrgFl );
}

/* Get strings, etc. */
if (ch == '\0')
{
    /* Get contents of string. */
    bf = sym->str;
    for (i = 0; i < MAX_STR; i++)
        if ((ch = fgetc( PrgFl )) != '\0' && ch != EOF)
            *bf++ = ch;
    else
        break;
    *bf = '\0';

    /* Return string token. */
    strcpy(word, sym->str);
    return( STRING );
}
else if (ch >= '0' && ch <= '9')
{
    auto    long    no;

    /* Establish a pointer to the word buffer and unget the
     * numeric character for re-reading.
     */
    bf = word;
    ungetc(ch, PrgFl);

    /* Get number, time or date. */
    if ((ch = GetNo(&bf, &no)) == ':')
    {
        /* Getting time, not number. */
        *bf++ = ch;
    }
}

```

_miniEd Editing Tools

The maximum solution for add-in text editing
Now with 'C' or Assembler source code

The best cross platform, cross language editing tools just got better. If you need to integrate text editing functions into your applications, the _miniEd family of editing products provides the answer. Handle document entry, pop-up windows, memo fields or any other formatted text requirement with a simple call to a _miniEd function.

Cross Platform: DOS / WINDOWS / UNIX

Cross Language: 'C' / BASIC / PASCAL /
ASSEMBLER / MODULA 2

Features: "Format-on-the-fly" word wrap technology
All standard editing functions supported
VIEW mode for formatted text display
Works with any size screen window
Full source code provided

Tools: _miniEd - RAM based editing in a 5K package
_miniEd Toolkit - File or RAM based editing
NEW _miniWords - 150,000 words & support routines
Coming In May _miniSpell - add-in spell checking

At SSDI, we don't take the word "mini" lightly. All _miniEd tools are compact, tightly structured modules that won't impose a size or performance overhead on your applications. The tools are priced from \$99.00 to \$349.00. To order or obtain more information call (708/778-6060) or FAX (708-778-6063). VISA and MC accepted. Please specify 'C' or assembler source when ordering.

Strategic Software Designs, Inc.
Software Tools and Custom Development
6S235 Steeple Run Drive - Suite 12B - Naperville, IL 60540
◆ Request 387 on Reader Service Card ◆

WCSC Professional Development Tools

COMM-DRV

- Professional serial communications library.
- Device Driver/Int21, Int14, FOSSIL
- De-Installable TSRw/INT14, FOSSIL
- 25+ Ports, Irq Sharing, remap ports
- Hdw./Softw. Handshaking/115 kbaud
- 8250/16450/16550 (Auto Detection)
- Desqview, Windows 3.0, pcAnywhere Device/TSR/Libraries/Source

\$89.95

COMM-LOG

- Background file transfer TSR
- Xmodem(1k)/Ymodem(Batch/G)
- Background logging to disk TSR
- Application & Command Line Interface
- File transfer/logging on separate ports / separate files concurrently. Easy to use.
- TSRs are de-installable
- Requires COMM-DRV

TSR & File Transfer Sources
\$89.95

SAVE!!! Buy 2 products for \$159.95, 3 for \$225.95, 4 for \$275.95

MTASK

- Preemptive Multitasking TSR)
- Allow creation of several threads of execution in your application.
- Threads may make DOS calls.
- Threads may sleep or be awakened
- Threads may have different time slice
- Intertask message support

TSR and Examples \$89.95

SCRN-INPUT

- Professional screen & keyboard library
- Create screens on monitor or over serial ports
- Save/Restore/Scroll screen areas
- Complete cursor & keyboard control
- Data driven input field screens
- Absolutely no royalties
- Extreme ease of use/Many examples

Library and Sources \$89.95

4 Port Serial Card w/16450 \$110 w/16550 \$170



1-800-966-4832

Willie's Computer Software Co.
2470 S. Dairy Ashford, Suite 188
Houston, Texas 77077

TEL (713)-498-4832
FAX (713)-568-3334
BBS (713)-568-6401

VISA/MC/AMEX/PO/COD/CHECK

24 Hours/Day 7 Days/Week

◆ Request 322 on Reader Service Card ◆

Listing 4 — Cont'd

```

sym->ftime.ti_hour = (unsigned char) no;
sym->ftime.ti_hund = (unsigned char) 0;

/* Get minutes. */
if ((ch = GetNo(&bf, &no)) == ':')
{
    /* Save minutes. */
    *bf++ = ch;
    sym->ftime.ti_min = (unsigned char) no;

    /* Get seconds. */
    if ((ch = GetNo(&bf, &no)) == '.')
    {
        *bf = '\0';
        ParseErr( "Hundredths of seconds not allowed in "
                  "time expressions" );
    }
    sym->ftime.ti_sec = (unsigned char) no;
}
else
{
    /* No seconds to get. */
    sym->ftime.ti_min = (unsigned char) no;
    sym->ftime.ti_sec = (unsigned char) 0;
}

/* This is a time. */
tkn = TIME;
}
else if (ch == '/')
{
    /* Getting date, not number. */
    *bf++ = ch;
    sym->fdate.da_mon = (char) no;
}

```

smx simple multitasking executive

Cut Embedded Programming Cost!

Are you reinventing the wheel? ... trying to write your own kernel? High-performance microprocessors, low-cost memory, and *smx* have made this unnecessary. *smx* costs a small fraction of what "home-brew" kernels cost to develop. Besides the programming time to write one, debugging is always harder than expected. And, when the home-brew is done, capabilities are usually disappointing. So what have you gained?

smx is a general-purpose, carefully-designed, reliable kernel. It has been in use for 3 years and already has a large base of satisfied customers. *What home-brew kernel can match this?*

In addition, *smx* does extensive error checking and features a powerful task debugger, called *smxProbe*. *smx* also has well-written, accurate manuals, and it's easy to learn and use. ... Leave the kernel writing to the experts — buy *smx*.

CALL OR FAX TODAY FOR FULL DETAILS

MICRO **pd** DIGITAL

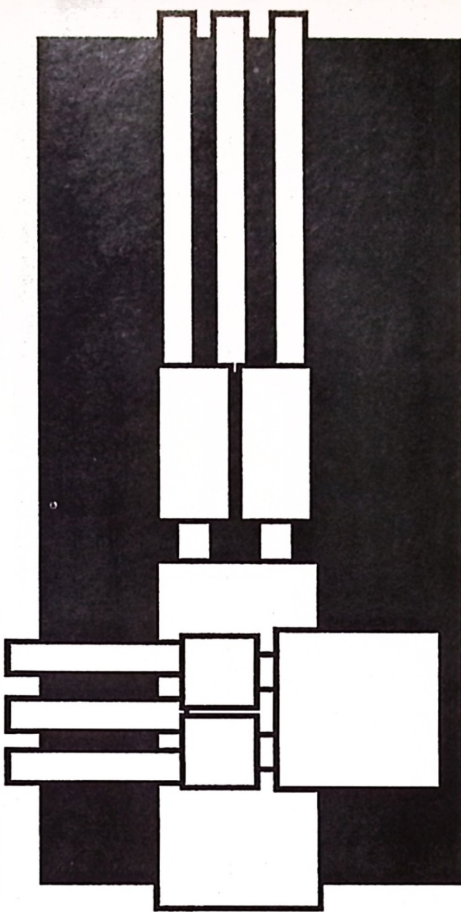
1-800-366-2491

6402 Tulagi St.

Cypress CA 90630-5630

FAX 714-891-2363

◆ Request 202 on Reader Service Card ◆



Listing 4 — Cont'd

```

/* Get day. */
if ((ch = GetNo(&bf, &no)) == '/')
{
    /* Save character. */
    *bf++ = ch;
    sym->fdate.da_day = (char) no;

    /* Get year. */
    ch = GetNo(&bf, &no);
    if (no > 1980L)
        no -= 1980L;
    else if (no > 80L && no < 100L)
        no -= 80L;
    else
    {
        *bf = '\0';
        ParseErr( "Error, bad year value in date expression." );
    }
    sym->fdate.da_year = (int) no;
}
else
{
    *bf = '\0';
    ParseErr( "Missing year in date expression" );
}

/* This is a date. */
tkn = DATE;
}
else
{
    /* Just an integer constant. */
    sym->no = no;
    tkn = NUMBER;
}

```

Real-Time Multitasking

for Microsoft C, Borland C, Turbo Pascal

Develop Real-Time Multitasking Applications with RTKernel!

RTKernel is a professional, high-performance, low-cost real-time multitasking kernel. It runs under MS-DOS and supports Microsoft C, Borland C++, Turbo C++, and Turbo Pascal. RTKernel is a library you can link to your application. It lets you run several functions / procedures as parallel tasks. Join hundreds of satisfied customers who've been using RTKernel in the past three years. RTKernel offers these advanced features:

- pre-emptive, event- / interrupt-driven scheduling
- number of tasks only limited by RAM
- task-switch time of approx. 12 µsecs on a 33-MHz 486
- performance independent of the number of tasks
- use different priorities (changeable at run-time) to control your tasks
- time-slicing can be activated
- programmable timer interrupt rate
- high-resolution timer (approx. 1 µsec)
- inter-task communications using semaphores, mailboxes, and message-passing
- call DOS from any number of tasks without re-entrance problems!
- activate tasks out of interrupt handlers
- supports math coprocessor and emulator
- supports COM1 to COM4
- extensible to any number of COM ports
- interrupt handlers for keyboard and COM ports included with source code
- resident multi-tasking applications (TSRs)
- keyboard, hard disk, floppy disk idle times usable by other tasks
- runs under MS-DOS 3.0 to 5.0, DR-DOS, Novell, etc., or without operating system
- ROMable
- supports CodeView and Turbo Debugger
- full source code available
- no run-time royalties
- free technical support by phone or fax

RTKernel-C (MSC 6.0, TC++ 1.0, BC++ 2.0) ... \$445 (for source code, add \$375)
RTKernel-Pascal (TP 5.0, 5.5, 6.0) \$375 (for source code, add \$325)

For international orders, add \$30 for shipping and handling.
Visa, check, bank transfer, or COD (in Europe) accepted.

Please call or fax for our free demo disk.

On Time
MARKETING
KARSTEN PETERSEN

Karolinenstrasse 32 · D-2000 Hamburg 36 · Germany
Phone +49 - 40 - 43 74 72 · Fax +49 - 40 - 43 51 96

◆ Request 101 on Reader Service Card ◆

QNX: Preferred by Real-Time, Performance Programmers

DynaDos:

Integrate favorite DOS applications into multi-user, multi-tasking QNX environment via direct Arcnet connection. Use QNX system as file server for DOS machines and as host for multi-user applications.

- New TCP/IP
- QNX Windows and associated tools
- C86 and C++ Compilers
- Flexlint source code analyzer
- Edit 2000 for professional programming
- Macro Assembler
- QWC windows toolkit, Freeze, QWEdit, Scholar
- File management utilities
- Menu utilities

Call for Catalogue of QNX software and hardware

T&T Computer Products, Inc.
QNX Specialists
P.O. Box 14010, Tulsa, OK 74159 USA
Tel. 918/663-1879 Fax 918/663-4054
VISA/MC

QNX® is a registered trademark of Quantum Software Systems Ltd.
DynaDos™ and DynaTerm™ are copyrights of T&T Computer Products, Inc.

◆ Request 247 on Reader Service Card ◆

Listing 4 — Cont'd

```

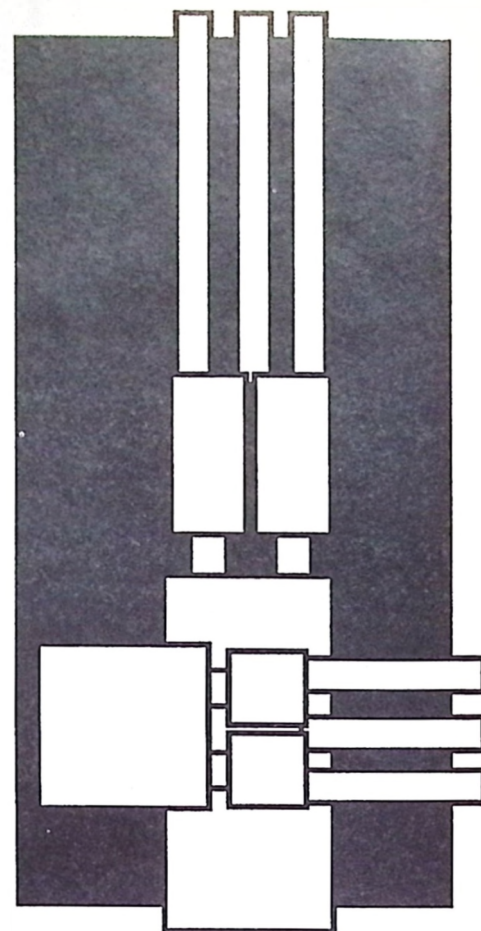
/* Return the unused character. */
*bf = '\0';
ungetc(ch, PrgFl);
return( tkn );
}
else if (ch == EOF)
    return( EOF );

/* Call the trie search routine to return the next token, EOF
 * or NOT_FND. If not found, print an error and quit.
 */
if ((tkn = TrieSrch(TO, ch, word)) == NOT_FND)
{
    /* Illegal first character in word. */
    if ( *PrvWd )
        fprintf(stderr, "Parse - Error in Line: %d, cannot identify "
            "word after '%s'.\n", LnNo + 1, PrvWd);
    else
        fprintf(stderr, "Parse - Error in Line: %d, cannot identify "
            "first word in file.\n", LnNo + 1);
    exit( -1 );
}
else if (tkn == 0)
{
    /* Illegal word. */
    fprintf(stderr, "Parse - Error in Line: %d, cannot identify "
        "word '%s'.\n", LnNo + 1, word);
    exit( -1 );
}
strcpy(PrvWd, word);

/* Return the token found. */
return( tkn );
}

/* End of File */

```



ANSI SQL DBMS FOR PROGRAMMERS, ONLY \$195

Supports - Turbo C • Turbo Pascal • Microsoft C • QuickBASIC • MicroFocus COBOL

All Compilers are trademarks of their respective Manufacturers.

OCELOT SQL gives you everything you expect from a SQL Database. It is:

SMALL

(needs only 320 KB RAM)

FAST

(written in assembly language)

DB2 COMPATIBLE

(conforms to industry standard)

FLEXIBLE

(runs embedded and stand-alone) and It includes REFERENTIAL INTEGRITY.



single user for Windows:	\$395
ditto with unlimited runtime:	\$1090
single user for basic, Pascal, C:	\$195
ditto with unlimited runtime:	\$695
multi-user for all languages:	\$850
ditto with unlimited runtime:	\$1850

Shipping & Handling:

USA: \$5.00 • CANADA/MEXICO: \$10.00 • Other: \$15.00 - California add tax.

Siener Soft

(USA) 6540 Lusk Blvd., Suite C-166 San Diego, CA 92121

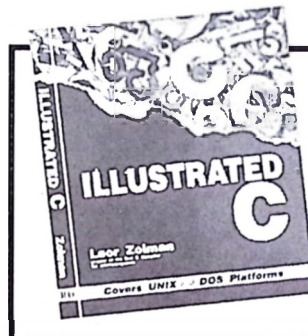
Phone: (619) 452-2265 • Fax: (619) 452-0145

(Germany) Phone: ++49-6126-59 50 • Fax: ++49-6126-5 10 85

(France) Phone: ++33-1-47 81 10 11 • Fax: ++33-1-42 42 37 10

BUILD A MINI-DATABASE SYSTEM WITH . . .

JUST RELEASED



Illustrated C
by **Leor Zolman**

CUJ columnist and author of BDS C

Discover the WHY and HOW of application design and development in C. Explore the construction of several different applications from start to finish. Chapter after chapter you'll develop your skills through in-depth tutorial and detailed code.

Only **\$29⁹⁵**

(**\$39⁹⁵** with disk)

R&D
publications, inc.

CALL TODAY!

913-841-1631

FAX 913-841-2624



◆ Request 479 on Reader Service Card ◆

◆ Request 495 on Reader Service Card ◆

Data Structures

Part 11: Yet More on Stacks

Handling Multiple Stacks of the Same Type

Last month's column described how to implement a stack and to use *push* and *pop* functions on it. However, it would be a waste to have a different set of *push* and *pop* functions for each stack. This month I will discuss how to share the code.

Consider Listing 1. An object of type *stack* contains the current context of a given stack. This context includes the stack's name (for debugging or trace-back purposes), the base address of the stack, the stack's size and its current stack pointer. The stack descriptors *stack1*, *stack2*, and *stack3* therefore describe the three different *int* stacks. The first stack is stored in a global array, the second in a file scope static array, while the third is allocated at runtime using *malloc*. In short, the stack management functions don't know and don't care where the stacks reside.

Listing 2 tells *push* and *pop* which stack to use but the notation is not particularly unwieldy. Listing 3 could be made a little bit cleaner by passing the stack descriptor by value, but that could be just a little more expensive since the descriptor is a structure. The output produced is shown in Figure 1.

Handling Multiple Stacks of Different Types

Can this idea be extended to manage stacks of different types? The answer is a qualified *yes*. One way could be to use generic pointers, as shown in Listing 4. Here, a *void* pointer is used to hold the stack's base address. This also requires an extra member to indicate the type of elements in any given stack, as in

```
void push(stack *, void *);  
void *pop(stack *);
```

The interface to *push* and *pop* gets very messy, however. Since an object of arbitrary type cannot be passed by value, it must be passed by assigning it to a named object and then passing that object by address, as shown in Listing 5. (This eliminates the ability to push a constant directly.) This is possible since all data pointer types are compatible with *void **. Similarly, an arbitrary typed value can't be returned, so the address is returned instead.

To use the value returned by *pop* you must use an explicit cast as well as a dereference since *pop* returns a pointer (see Listing 6). In fact, since *pop* returns the address of the object just popped from the stack, if you don't dereference the pointer returned immediately, the location it points to will be overwritten by the very next push and the value popped will change.



Rex Jaeschke is an independent computer consultant, author, and seminar leader. He participates in both ANSI and ISO C Standards meetings and is the editor of The Journal of C Language Translation, a quarterly publication aimed at implementors of C language translation tools. Readers are encouraged to submit column topics and suggestions to Rex at 2051 Swans Neck Way, Reston, VA 22091 or via UUCP at rex@aussie.com.

Listing 1

```
#include <stdio.h>
#include <stdlib.h>

#define NUMELEMENTS(x) (sizeof(x)/sizeof(x[0]))

typedef struct {
    const char *stack_name;
    int *pstack;
    size_t stack_ptr;
    size_t max_stack;
} stack;

int array1[1];
static int array2[30];

stack stack1 = {"stack1", array1, 0, NUMELEMENTS(array1)};
stack stack2 = {"stack2", array2, 0, NUMELEMENTS(array2)};
stack stack3 = {"stack3", NULL, 0, 0};

/* End of File */
```

Listing 2

```
void push(stack *, int);
int pop(stack *);

main()
{
    int size = 50;

    stack3.pstack = malloc(size * sizeof(int));
    if (stack3.pstack == NULL) {
        printf("Can't allocate space for stack3\n");
        exit(1);
    }

    stack3.max_stack = size;

    push(&stack1, 10);
    push(&stack1, 12);
    push(&stack2, 15);
    push(&stack3, 20);
    printf("stk1: %d\n", pop(&stack1));
    printf("stk2: %d\n", pop(&stack2));
    printf("stk3: %d\n", pop(&stack3));
    printf("stk3: %d\n", pop(&stack3));

    return 0;
}

/* End of File */
```

Listing 3

```
void push(stack *st, int value)
{
    if (st->stack_ptr == st->max_stack)
        printf("Stack %s is full\n", st->stack_name);
    else
        st->pstack[st->stack_ptr++] = value;
}

int pop(stack *st)
{
    if (st->stack_ptr == 0) {
        printf("Stack %s is empty\n", st->stack_name);
        return 0;
    }

    return st->pstack[--st->stack_ptr];
}

/* End of File */
```

The source for *push* and *pop* is far from pretty. Since it cannot perform arithmetic (via subscripting) on *void* pointers, you must explicitly provide the scaling factor.

Using Unions

Another way to look at the problem is rather than have stacks of different types, have one stack that can handle objects of different types. You can achieve this via a union, as shown in Listing 7.

Each entry is a union of all the possible types along with a flag member that indicates which type this entry currently represents. We push nodes by value and return them by value using simple and obvious notation. If you try to pop from an empty stack, instead of complaining, *pop* returns a copy of a local node that has a special type field value of *Error*.

Next, consider Listing 8. Note the interesting controlling expression in the *while* loop — it uses the comma operator in an effective manner.

The stack management functions presented in Listing 9 are quite straightforward.

Opposing Stacks

A stack grows and shrinks from one end only so it is possible to have two stacks based at opposite ends of an array with their tops growing toward each other. This can save space if both stacks don't grow very large at the same time. However, when one is smaller, the other can grow larger and vice-versa.

The *dump_stack* function in Listing 10 allows us to see the contents of the underlying array as the two stacks grow and shrink. Note that it pops in a different order than it pushes, so the operations are staggered.

Figure 1

```
Stack stack1 is full
stk1: 10
stk2: 15
stk3: 20
Stack stack3 is empty
stk3: 0
```

Listing 5

```
main()
{
    int size = 50;
    int vali = 10;
    long val1 = 456789;
    double vald = 123.456;

    stack3.pstack = malloc(size * sizeof(double));
    if (stack3.pstack == NULL) {
        printf("Can't allocate space for stack3\n");
        exit(1);
    }

    stack3.max_stack = size;

    push(&stack1, &vali);
    push(&stack2, &val1);
    push(&stack3, &vald);
    printf("stk1: %d\n", *((int *)pop(&stack1)));
    printf("stk2: %ld\n", *((long *)pop(&stack2)));
    printf("stk3: %f\n", *((double *)pop(&stack3)));

    return 0;
}

/* End of File */
```


Listing 4

```
#include <stdio.h>
#include <stdlib.h>

#define NUMELEMENTS(x) (sizeof(x)/sizeof(x[0]))
#define INT 1
#define LONG 2
#define DOUBLE 3

typedef struct {
    const char *stack_name;
    void *pstack;
    const int type; /* type of entries */
    size_t stack_ptr;
    size_t max_stack;
} stack;

int array1[10];
static long array2[30];

stack stack1 = {"stack1", array1, INT, 0, NUMELEMENTS(array1)};
stack stack2 = {"stack2", array2, LONG, 0, NUMELEMENTS(array2)};
stack stack3 = {"stack3", NULL, DOUBLE, 0, 0};

/* End of File */
```

Listing 6

```
void push(stack *st, void *pvalue)
{
    if (st->stack_ptr == st->max_stack)
        printf("Stack %s is full\n", st->stack_name);
    else
        switch (st->type) {
            case INT:
                ((int *)st->pstack)[st->stack_ptr++] = *(int *)pvalue;
                break;
            case LONG:
                ((long *)st->pstack)[st->stack_ptr++] = *(long *)pvalue;
                break;
            case DOUBLE:
                ((double *)st->pstack)[st->stack_ptr++] = *(double *)pvalue;
                break;
        }
}

void *pop(stack *st)
{
    if (st->stack_ptr == 0) {
        printf("Stack %s is empty\n", st->stack_name);
        return 0;
    }

    switch (st->type) {
        case INT:
            return &((int *)st->pstack)[-st->stack_ptr];
            break;
        case LONG:
            return &((long *)st->pstack)[-st->stack_ptr];
            break;
        case DOUBLE:
            return &((double *)st->pstack)[-st->stack_ptr];
            break;
    }
}

/* End of File */
```

Confusing Code?

```
#include <stdio.h>
text_count() {int c, nlines, nwords, nchars, inword;
inword=NO; nlines=nwords=nchars=0; while((c
=getchar()) != EOF) {++nchars; if (c=='\n')
++nlines; if ((c==' ') || (c=='\n')) inword=NO; else
if (inword==NO) {inword=YES; ++nwords;} printf(
"%d %d %d\n", nlines, nwords, nchars);}
```

C It Your Way!

```
#include <stdio.h>
text_count ()
{
    int c, nlines, nwords, nchars, inword;

    inword = NO;
    nlines = nwords = nchars = 0;
    while ((c = getchar()) != EOF) {
        ++nchars;
        if (c == '\n')
            ++nlines;
        if ((c == ' ') || (c == '\n'))
            inword = NO;
        else if (inword == NO) {
            inword = YES;
            ++nwords;
        }
    }
    printf ("%d %d %d\n", nlines, nwords, nchars);
}
```

(One of an infinite number of styles)

NOW with C-Clearly,™ format C source code exactly the way you want it. C-Clearly's context sensitive analysis will format any C program in your own personal or corporate style.

EASY to use, C-Clearly's style templates are a snap to modify, since they resemble C source code you edit into your preferred format. Templates are included for several common styles as well as standard K & R.

LISTINGS can also be created with function names and comments highlighted for improved readability. Listing options include line numbers, headers and/or footers and flow lines.

IDEAL for making obtuse code clear. Allows all of your source code to be presented in a consistent format of your choosing. Also great for code walkthroughs and final documentation listings.

WORKS with all IBM PC, XT, AT, PS 2 and compatibles, with 512K RAM. Automatically processes all include files and pre-processor statements. ANSI-C compatible. Not copy protected.

C-Clearly \$129.⁹⁵

Shipping & Handling USA \$5. Canada Mexico \$10. Other Countries \$15. CA Residents add sales tax. Visa MasterCard CDD accepted

For orders or information call:

1-800-648-8266



V Communications Incorporated
4320 Stevens Creek Blvd, Suite 275-CU
San Jose, CA 95129
(408) 296-4224

◆ Request 179 on Reader Service Card ◆

Listing 7

```
#include <stdio.h>

enum node_type {Error, Char, Int, Double, String};

typedef struct {
    enum node_type type;
    union {
        char c;
        int i;
        double d;
        char *s;
    } value;
} node;

void push(node);
node pop(void);

/* End of File */
```

The stack in Listing 11 can only handle four elements. This helps you monitor the progress as elements are pushed and popped. Clearly, it would be larger in a real application. Of course, the bases of the two stacks are at either end of the underlying array and in one stack the stack pointer increments as we push and in the other it decrements. Stack overflow is detected when the two stack pointers bump into each other. Note that it's OK if both of them point to the same element since that last free element is available to which ever stack can use it first.

The two versions of *push* and *pop* in Listing 12 are different enough that it is not obvious you can write a single version

SIMPLE TOOLKIT FOR X-WINDOWS

The "unofficial toolkit" for
programmers who don't have
time to waste.

Follows OSF/Motif™ user
interface style. Full C source
and interface builder included.

\$229 for 1-2 users, corporate
site license \$458. Call today for
risk-free 60 day trial.

STROM SYSTEMS, INC.

18666 Redmond Way O-2118, Redmond, WA 98052

Phone (206) 836-2241

Listing 8

```
main()
{
    node n;

    n.value.d = 9.87;
    n.type = Double;
    push(n);

    n.value.s = "some text";
    n.type = String;
    push(n);

    n.value.i = 123;
    n.type = Int;
    push(n);

    n.value.c = 'A';
    n.type = Char;
    push(n);

    while (n = pop(), n.type != Error) {
        switch (n.type) {
            case Char:
                printf("Char = %c\n", n.value.c);
                break;
            case Int:
                printf("Int = %d\n", n.value.i);
                break;
            case Double:
                printf("Double = %f\n", n.value.d);
                break;
            case String:
                printf("String = %s\n", n.value.s);
                break;
        }
    }

    return 0;
}

/* End of File */
```

Figure 2

Stack contains:	0	0	0	0	sp1 = 0, sp2 = 3
Stack contains:	10	0	0	0	sp1 = 1, sp2 = 3
Stack contains:	10	0	0	12	sp1 = 1, sp2 = 2
Stack contains:	10	15	0	12	sp1 = 2, sp2 = 2
Stack contains:	10	15	34	12	sp1 = 2, sp2 = 1
Stack 1 is full					
Stack contains:	10	15	34	12	sp1 = 2, sp2 = 1
Stack 2 is full					
Stack contains:	10	15	34	12	sp1 = 2, sp2 = 1
stack 1: 15					
Stack contains:	10	15	34	12	sp1 = 1, sp2 = 1
stack 1: 10					
Stack contains:	10	15	34	12	sp1 = 0, sp2 = 1
Stack 1 is empty					
stack 1: 0					
Stack contains:	10	15	34	12	sp1 = 0, sp2 = 1
stack 2: 34					
Stack contains:	10	15	34	12	sp1 = 0, sp2 = 2
stack 2: 12					
Stack contains:	10	15	34	12	sp1 = 0, sp2 = 3
Stack 2 is empty					
stack 2: 0					
Stack contains:	10	15	34	12	sp1 = 0, sp2 = 3

Listing 9

```
#include <stdio.h>

#define STACK_SIZE 10

static node stack[STACK_SIZE];
static size_t stack_ptr = 0;

void push(node n)
{
    if (stack_ptr == STACK_SIZE)
        printf("Stack is full\n");
    else
        stack[stack_ptr++] = n;
}

node pop(void)
{
    static node error_node = {Error, 0};

    if (stack_ptr == 0)
        return error_node;
    else
        return stack[--stack_ptr];
}

/* End of File */
```

that is both elegant and efficient. The output produced is shown in Figure 2.

Listing 10

```
#include <stdio.h>

void push1(int);
void push2(int);
int pop1(void);
int pop2(void);
void dump_stack(void);

main()
{
    dump_stack(); push1(10);
    dump_stack(); push2(12);
    dump_stack(); push1(15);
    dump_stack(); push2(34);
    dump_stack(); push1(99);
    dump_stack(); push2(65);
    dump_stack(); printf("stack 1: %d\n", pop1());
    dump_stack(); printf("stack 1: %d\n", pop1());
    dump_stack(); printf("stack 1: %d\n", pop1());
    dump_stack(); printf("stack 2: %d\n", pop2());
    dump_stack(); printf("stack 2: %d\n", pop2());
    dump_stack(); printf("stack 2: %d\n", pop2());
    dump_stack();

    return 0;
}

/* End of File */
```

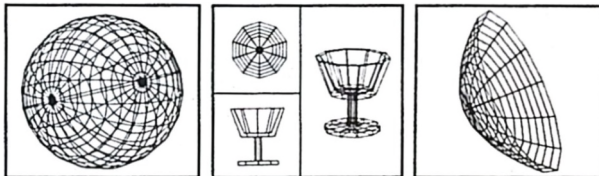
Note that even after a value is popped from a stack it still remains there — only the stack pointer is adjusted. This is exactly what happens when a C function returns; the values of

CAD/CAM/CAE Graphic Developers!!

TGL PROFESSIONAL

(Formerly TurboGeometry Library 3.0)

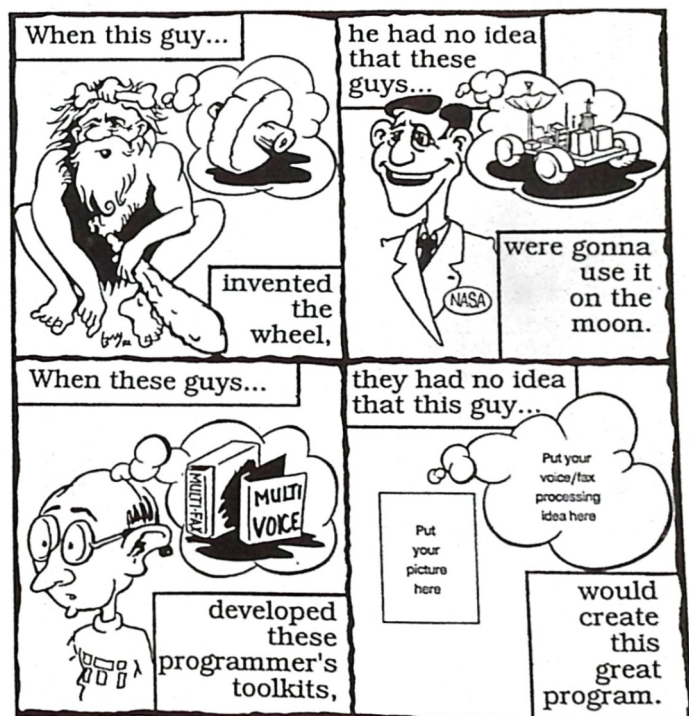
"The Ultimate CAD/CAM/CAE Graphics Programming Engine"



Now available for your programming needs. Over 600 2D and 3D geometric routines for CAD/CAM/CAE and graphics applications. Includes NURBS, Surfacing, Hidden Line, Perspective, Rotation, Scaling, Clipping, Polygon (Int/Union/Diff) and more. Full range of DXF input/output routines for importing and exporting 2D and 3D geometry between DXF compatible systems. Supports MSC and BGI graphics libraries. Draw 2D and 3D objects generated by the library and still maintain device independent geometric routines. TGL Professional also includes a general utility library (TGL Utilities) that perform list management, sorting, DXF in/out. Also has a new graphics user interface (GUI) which gives you the ability to create dialog boxes, pull-down menus and pop-up windows for easy input/output of data and much more. TGL Professional comes with full source code, manuals and example programs. *Compatibility:* MSDOS, PC's *Language:* MSC and Turbo C/C++. TGL 2D Library, TGL 3D Library and TGL Utilities, that make up TGL Professional, may be purchased separately. A technical white paper on TGL Professional is available upon request. Prices: TGL Professional \$500., TGL 2D \$300., TGL 3D \$300., & TGL Utilities \$200. Call or Fax for foreign pricing.

Disk Software Inc. 2116 E. Arapaho Rd., Suite 487
Richardson, TX, USA 75081. Phone/Fax 214-423-7288, 1-800-635-7760

◆ Request 116 on Reader Service Card ◆



ITI Logiciel
4165 A St-Denis
Montréal, Québec
Canada
H2W 2M7

CALL NOW
Tel (514) 847-2240
Fax (514) 847-2242



Eliminate all your listing errors

...with one phone call.
Don't waste your time
typing in long code listings.
Let us do the work for
you, --save yourself from
the hassle and the errors.

only

\$30*

for a full year
(12 disks) of code
Save 50% off
individual disks.

or

\$59⁹⁵*

for a full year
of the magazine
and the disks.

**Now get the C Users Journal
and all its code on disk
in your mail box every month.**

Call (913) 841-1631 today!

Subscriptions must be prepaid and
are available on 5 1/4 or 3 1/2 MS DOS
format only.

*Call for foreign prices.

The **C** Users Journal™

it's automatic variables are still on the stack but are outside the bounds of the newly adjusted stack pointer. They remain intact until that part of the stack is overwritten for some other purpose. □

Listing 11

```
#include <stdio.h>

#define STACK_SIZE 4

static int stack[STACK_SIZE];

static size_t stack_ptr1 = 0;
static size_t stack_ptr2 = STACK_SIZE - 1;

/* End of File */
```

Listing 12

```
void push1(int value)
{
    if (stack_ptr1 > stack_ptr2)
        printf("Stack 1 is full\n");
    else
        stack[stack_ptr1++] = value;
}

void push2(int value)
{
    if (stack_ptr1 > stack_ptr2)
        printf("Stack 2 is full\n");
    else
        stack[stack_ptr2--] = value;
}

int pop1(void)
{
    if (stack_ptr1 == 0) {
        printf("Stack 1 is empty\n");
        return 0;
    }

    return stack[--stack_ptr1];
}

int pop2(void)
{
    if (stack_ptr2 == STACK_SIZE - 1) {
        printf("Stack 2 is empty\n");
        return 0;
    }

    return stack[++stack_ptr2];
}

void dump_stack(void)
{
    int i;

    printf("Stack contains: ");
    for (i = 0; i < STACK_SIZE; ++i)
        printf("%4d", stack[i]);

    printf("\tsp1 = %lu, sp2 = %lu\n",
        (unsigned long)stack_ptr2);
}

/* End of File */
```


C Express: 250+ Ready-To-Run Assembly-Language Routines for Turbo C, Microsoft C, and QuickC

*C Express: 250+ Ready-To-Run
Assembly-Language Routines for
Turbo C, Microsoft C and QuickC*
Robert Jourdain
Brady Books, 1989
ISBN 0-13-933185-9

Reviewed by Stephen Patten

C Express: 250+ Ready-To-Run Assembly-Language Routines for Turbo C, Microsoft C and QuickC, a book and accompanying diskettes, was written (both the assembly source code and text) by Robert Jourdain and published by Brady Books, a division of Simon & Schuster, Inc. The copy I reviewed was dated 1989.

Companion Diskettes

Two 5.25" diskettes accompany the 412-page book. One diskette contains the eight library files which house the 250 plus routines in object code form along with eight parallel header files for function prototyping. The other diskette contains the assembly language source of the routines in compressed form. (You can exchange the 5.25" disks for 3.5").

You install the libraries and header files on a hard drive by a simple copy, and then call routines as you would any C function inside source code. A global variable is also set which is used by the compiler to configure the correct memory model.

Book Contents

Chapter 1 provides a short introduction to using the library. Chapters 2-9 document the library files included on the companion disk. Chapter 10 includes a discussion on writing your own assembly subroutines to link with C programs. This is followed by indices of general topics, the routines, and their object files.

The routines or functions are documented pretty much as you would expect, each described in terms of its purpose, parameters, globals required, kinds of errors checked, and relevant peculiarities. Examples of calls are also amply provided.

The book also presents background information on groups of routines. For example, functions that implement expanded memory are preceded by a description of the LIM specification, not in great detail, but enough for the programmer to understand what the routines do and why.

Stephen Patten is a senior analyst with the Lincoln Savings Bank in New York and teaches C at New York University. He can be reached at (516) 932-3484.

Interestingly, video routines which can be implemented in either ROM BIOS or memory-mapped form are presented both ways, with the ROM BIOS routines suffixed with an underscore and lower case *b*. Depending upon the application, the programmer can choose between speed and portability.

Overview of Functions

The equipment configuration routines access the usual details of installed hardware and are only notable in that they eliminate the bit manipulations required when using the compiler library to extract the same information.

The memory management routines test for extended and expanded memory, and allow programming of expanded memory applications. Pages can be allocated, switched and deallocated, and data exchanged between expanded memory and RAM quickly and easily. This can be a real plus to a C programmer restricted by the one megabyte limitation of the MS-DOS environment.

C Express routines, like *hex_to_char*, *hex_to_int*, *binary_to_char*, and *binary_to_int*, interpret such strings as numbers, producing values which can then be entered as numeric data in C programs. Combined with existing *stdlib.h*

utility functions, like *ltoa*, they can also be used to convert binary to hex, or vice versa, a nice library facility.

There are string manipulation routines to add or delete characters to or from strings, change string characters, and perform various substring operations. There are also keyboard routines that provide for fast operating system calls to check, read, clear, wait on and change characters, and scan codes in the keyboard buffer.

filter_in, for example, captures the next keystroke in the buffer then looks for the same character in a predefined table. If found, the keystroke is accepted; otherwise, it's rejected. A related routine, *filter_out*, does the opposite, rejecting a character found in the table.

key_pause allows a key to act like an ON-OFF switch. Hold it down and, let's say, a help screen appears. Release it, and the help screen disappears. This can replace long and complicated *if* and *switch* statements with cleaner, faster code.

Mouse routines, based on Microsoft's driver, position the mouse cursor, turn it on and off, capture a button press, report mouse positioning and control mouse motion to cursor movement. With *pixel_ratio*, you can set the exact movement ratio between the mouse and its cursor motion. With

Learn C++, OOP, and X from Best-Selling Author Naba Barkakati

Object-Oriented Programming in C++

A practical guide that explains the basic concepts of object-oriented programming (OOP) and shows how to apply OOP techniques in C++. Includes examples of DOS and Windows applications in C++. Get the code on a 3.5" DOS disk to learn C++ and OOP. Ideal for Borland C++. Book (695 pg) plus disk \$49.95 Book or Disk only \$29.95

X Window System Programming

Tutorial/reference for C programmers learning to program X with Xlib and the Motif widgets. Code available on a 3.5" diskette in MS-DOS or UNIX tar format. Book (782 pg) plus disk \$49.95 Book or Disk only \$29.95

Both books published in 1991 by SAMS, Carmel, IN.

VISA/MasterCard Orders Accepted

Call or send payment (money order or check in U.S. Dollars, or VISA/MasterCard account information) to:

L.N.B. SOFTWARE, INC.

2005 Aventurine Way, Silver Spring, MD 20904 USA

Voice: (703) 284-2009 FAX: (703) 276-0759

Shipping: \$3 per book in U.S., \$6 per book outside U.S. Maryland resident, add sales tax. Written orders, include VISA/MC expiration date.

◆ Request 198 on Reader Service Card ◆

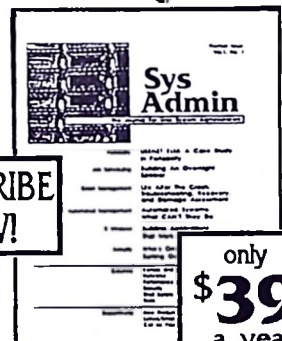
The Answer to All Your UNIX System Administration Questions

- Performance Tuning
- Shell Tools
- Kernel Resources
- Device Drivers
- Networks
- Security
- and more

**SUBSCRIBE
NOW!**

**Sys
Admin**

The Journal For UNIX System Administrators



only
\$39
a year
\$69 non US

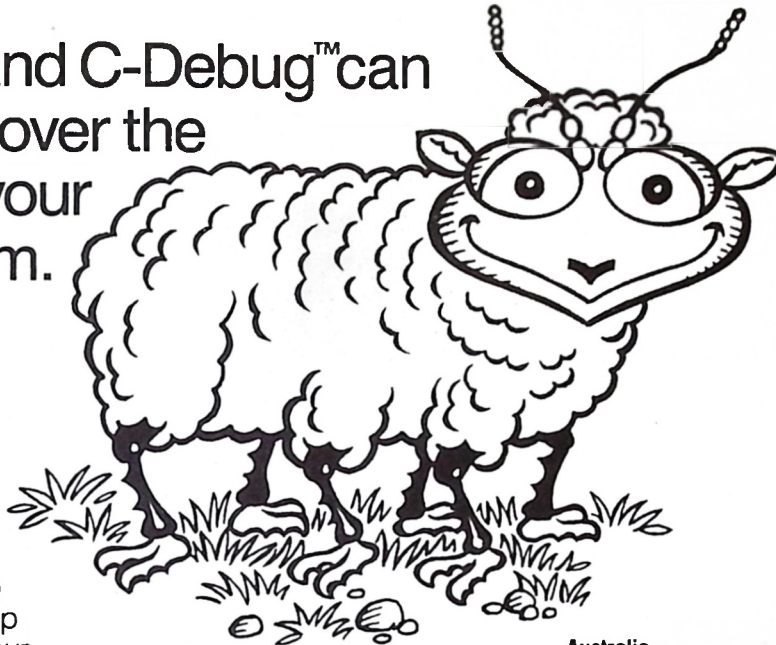
You get solid technical information, 6 times a year, full of ways to improve the performance and extend the capabilities of your system—whether it's a stand alone 386 with 3 users or a VAX with 300. A years worth of timely, effective tips and techniques in one resource. If you administer a UNIX system—SysAdmin can save you and your organization time and money.

All you need to keep your system
up and running — better than ever!

913-841-1631

Would you know one if you saw one?

C-Verify™ and C-Debug™ can
help you uncover the
bugs in your
program.



C-Verify™ and C-Debug™, two
utilities from Softran™, can help
you protect the integrity of your
C-language programs.

Announcing C-Verify, A Coverage Analyzer

C-Verify is Softran's newest
product. It is a coverage analyzer
for testing C-language programs.

C-Verify parses the source
code, then assigns a unique label
to every branch, inserting a
function call that writes a record
to a file each time a branch is
taken. When you run your tests
(in batch mode, if you like), you
receive a report documenting
which branches were taken, and
which were not. This gives you an
opportunity to revise your test
suite until you achieve 100
percent coverage.

C-Debug Gets Rave Reviews!

C-Debug, when used with C-Verify,
enables you to test and debug
every part of your program. Here's
what *Unix Review* had to say
about C-Debug:

*The name "C-Debug:
The Pointer Checker for All
Systems" describes this product
conveniently. It excels at finding
pointer problems in code, which
may arise due to initialization,
reassignment, or freeing of
variables and arrays. . . . C-Debug
is useful. Once it has trapped a
few of your errors, you will quickly
get in the habit of relying on it. . . .
The overhead C-Debug adds is
minimal, and its benefits are
considerable.*

Available for MS-DOS and UNIX

C-Verify and C-Debug can be run
at the same time and will work
with all C compilers. They are
available for both MS-DOS and
UNIX systems.

To find out more about
C-Verify and C-Debug, call
1-800-462-3932. Or in Canada,
call 1-708-505-3456. (A complete
listing of toll-free international
phone numbers follows)

Australia	0014-800-127-804
Belgium	11-9484
Denmark	8001-8410
Finland	9800-1-56196
France	05-90-2581
Ireland	1-800-55-8978
Japan (KDD)	031-11-2776
South Korea	08-1-800-929-8256
Luxembourg	0800-4297
Mexico (SA1, SA2, SA3 & SA4)	95-800-010-0862
Netherlands	06-022-1932
New Zealand	0800-44-0709
Norway	050-12-665
Spain	900-96-1208
Sweden	020-79-3372
Switzerland & Liechtenstein	155-0422
United Kingdom (BTI)	0800-89-4746
West Germany	0130-81-1838

Softran
CORPORATION



Softran Corporation
One Naperville Plaza, Suite 455
Naperville, IL USA 60563

`define_graphics_cursor`, you can set the precise pixel or "hot spot" inside the pattern of pixels that fall under the cursor display.

Screen control routines set the video adapter, position the cursor, set mono or color attributes, and control the writing of strings to the screen. For example, `clear_blink` changes the 7 bit in the attribute byte so that it controls the intensity of background colors rather than blinking, effectively doubling background colors in CGA text mode. `set_blink` restores the bit to its original interpretation.

`display` and `display_b`, unlike `printf`, are passed arguments to both position a string before displaying, and display with an attribute.

The graphics in C Express are limited to the CGA text mode. Like the screen control routines, there are both memory-mapped and BIOS versions. They draw the usual assortment of graphics figures as well as scroll sections of the screen either vertically or horizontally. Written in assembly, they run very fast.

The file routines are built around a data structure called `tree_array` which holds directory names and pointers to sub-directories. The routines work with the structure displaying a

directory tree or searching through the tree for a file. It's a little tricky going at first, but the book makes it all pretty clear.

Finally, a variety of printer output services are included. They simplify sending output to the printer inside a C program, do a certain amount of printer formatting, interpret embedded control codes in strings and implement the BIOS print screen function. `wrap_line` formats a string for printing in word wrapped form, while `justify_line` formats a string in right-justified, word wrapped form. The `prtsc` routine performs a screen dump, sending both text and graphics images to the printer.

In conclusion, the C Express package provides a collection of object code files and routines for calling within a C program. The routines are not unique in terms of their functionality but are a fairly standard collection that can be found in other off the shelf libraries. They are easy to set up and use, and they run fast. I was unable to find any bugs. The book provides good, well-organized documentation, and guidance for using the library files.

Particularly for new C programmers, the package would very likely make an excellent starter kit and base from which to build their own language libraries. □

Advanced System Building Blocks

CubiCalc®
The Third Wave in Intelligent Software

The OWL Neural Network Library
Powerful
Affordable
Flexible

Fuzzy Logic. Neural Networks.

- CubiCalc Fuzzy Logic Shell. The fastest way ever to learn or apply fuzzy logic. Optional callable runtime and fuzzy logic compiler.
- OWL Neural Networks. C-callable runtime libraries. Portable C source available. (from Olmsted & Watkins)

HyperLogic™

HyperLogic Corporation
1855 E. Valley Parkway, #210
Escondido, CA 92027 USA
Phone 619 / 746-2765
Fax 619 / 746-4089

◆ Request 161 on Reader Service Card ◆

Integrity is critical to our publication.

Integrity begins with ethical information gathering and reporting and extends to our advertising sales department as well. Advertisers can believe in the integrity of our circulation information — the basis for their advertising buying decisions.

That's why we subject our circulation records to independent verification by the largest and oldest circulation auditing organization in the world — the Audit Bureau of Circulations.

With verified reports from ABC, advertisers and choose us with absolute confidence.



**Audit Bureau of Circulations
Member**

It's Back!

It's official. Paul Vixie has indeed taken over the moderation of *comp.sources.unix*. He also has the assistance of two co-moderators, Mike Stump and Nick Lai, to help him prevent the kind of backlog that occurred recently. Submissions to the news group, and comments to the moderators should be sent to *unix-sources-moderator@pa.dec.com* or *decwrlunix-sources-moderator*. The queue has been cleared, and the backlog posted. However, it includes over six megabytes of sources in 37 separate postings. Here are just the highlights.

The first posting of the "new era" was *chop* from George Sicherman <gls@hrmso.att.com>. *chop* extracts selected fields or columns of lines from the specified files or standard input, and writes them to standard output. It is similar to *cut*, a tool from the UNIX Documenters Workbench. *chop* allows for variable field separators and for output of the fields in the order specified on the command line. *chop* is Volume 25, Issue 1.

Victor Abell <abe@mace.cc.purdue.edu> has obsoleted two of his older postings, *ofiles* and *fstat*, with a new version of *lsof*. LiSt Open Files displays the names of files opened by processes on selected UNIX systems. This new version in Volume 25, Issue 2, supports any file system based on the SunOS *vnode* model. This includes SunOS, AIX, HP-UX, NeXTStep, Dynix and some others. It understands most *vnode* extensions, NFS connections, FIFOs, multiplexed files, and UNIX and INET sockets.

Abell having obsoleted *ofiles*, Robert Ehrlich <ehrllich@margaux.inria.fr> then went and reworked the original *ofiles* and released *ofiles2* for Volume 25, Issue 72.

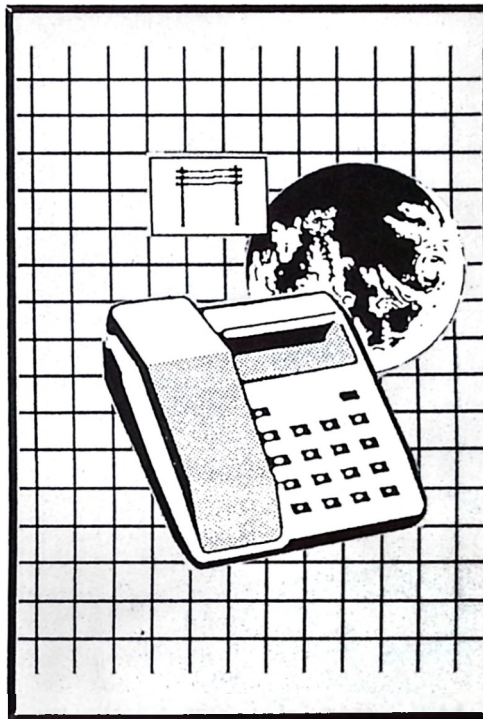
This new version is more portable, but is still designed for BSD derived UNIX systems. As with *lsof*, *ofiles2* displays the names of files that are being used by processes.

A major update to the "Threaded RN" Newsreader was issued by Wayne Davison <davison@boreland.com> for Volume 25, Issues 4-16. TRN is based on RN 4.4, and the posting includes both RN and the changes to make TRN from RN. TRN uses the References header to build a discussion thread out of the news group. The articles are then presented in discussion thread order.

One of the long promised postings, held in the queue for ages, is *ease* v3.5. *ease* is a decompiler that converts *sendmail.cf* into a language that is much easier to understand, and then allows editing of that intermediate language. It also includes a compiler to convert the *ease* language back into a *sendmail.cf* file. Considering how unreadable *sendmail.cf* files are, anything is an improvement. However, *ease* is a vast improvement. Bruce Barnett <barnett@crdgwl.ge.com> contributed *ease* for Volume 25 Issues 17-22.

Wen-King Su <wenking@vlsi.cs.caltech.edu> contributed *fsp*, one of the more interesting postings. It consists of a set of programs that implement a public-access

archive server similar to *anonymous-ftp*. While the actual code is probably not of much use to most CUJ subscribers, the concept and the implementation of that concept are a very well thought out lesson in client/server computing. It shows the tradeoffs between multiple servers, and a single stateless server. FSP was posted as Volume 25, Issues 24-26.



Sydney S. Weinstein, CDP, CCP is a consultant, columnist, author, and president of Datacomp Systems, Inc., a consulting and contract programming firm specializing in databases, data presentation and windowing, transaction processing, networking, testing and test suites, and device management for UNIX and MS-DOS. He can be contacted care of Datacomp Systems, Inc., 3837 Byron Road, Huntingdon Valley, PA 19006-2320 or via electronic mail on the Internet/Usenet mailbox *syd@DSI.COM* (*dsinc!syd* for those who cannot do Internet addressing).

Several of the newer UNIX shells provide interactive command line editing. The *tle* program from Robert C. Pendleton <bobp@hal.com> provides this same capability for any program. It runs the program as a subprocess so it does not need to modify the program, nor does it even need the source of the program it is servicing. It cannot provide a command history from program to program however. *tle* is Volume 25, Issue 29.

One of the nicer capabilities of the Bitnet sites is the *listserv* program provided to automate maintenance of mailing lists. Anastasios Kotsikonas <tasos@cs.bu.edu> has provided similar capabilities in his *listserv* v5.31 package posted in Volume 25, Issues 35-40. *listserv* provides support for *list*, *list-owner*, and *list-request* aliases for the mailing list and provides the back end programs for archives, moderated lists, peer lists, automated subscription, changes of address, and canceling of subscriptions.

I get spoiled by running on a bit-mapped screen with a good windowing system. Just writing this column I have several windows open, one for the column, one for the list of files to write about, and one containing the file I am reviewing. For those stuck on standard ASCII terminals Juergen Weigert <jnweiger@immd4.informatik.uni-erlangen.de> has released version 3 of his multi-session package, *screen3*, for Volume 25, Issues 41-48. It allows several different virtual screens on a single terminal using a short "hot key" sequence to switch between the sessions.

One of the more popular UNIX shells for interactive use has been *csh*, the c-shell. Over the years, many interactive ex-

tensions to *csh* were published as *tcsh*. These extensions used to require the source of *csh* as the base for the patches. Now with version 6, the full source of *tcsh* can be released. *tcsh* is a "kitchen sink" shell. It supports command line editing, command and file name completion, lists, manual lookup, job control, and has been ported to many different UNIX systems. For those that prefer *csh* to *ksh*, then *tcsh* will give you all the benefits of *ksh* with the *csh* syntax. The newest version, 6.01, was contributed by Christos Zoulas <christos@ee.cornell.edu> for Volume 25, Issues 54-71.

Another major update is the latest version of the Revision Control System. RCS v5.6, which is a very flexible source code librarian not only supports source files, but also can track changes to binary files. RCS v5.6 was contributed by Adam Hammer <hammer@cs.purdue.edu> for Volume 25, Issues 77-87. New in v5.6 are changes to fix security problems, efficiency changes for retrieving older versions, and the following of symbolic links instead of breaking them, and more reliable lock files under NFS.

Emmet Gray <fthood!gray@uxc.cso.uiuc.edu> has updated his *mtools* package to version 2.0.5. *mtools* allows UNIX systems read, write, and manipulate files on an MS-DOS filesystem (typically a diskette). It emulates the commands *ATTRIB*, *CD*, *COPY*, *DEL/ERASE*, *DIR*, *FORMAT*, *LABEL*, *MD/MKDIR*, *RD/RMDIR*, *COPY*, *REN/RENAME*, *TYPE*, and *COPY*. The *FORMAT* command only adds the MS-DOS file system to the diskette. It depends on the UNIX low-level format routines to actually low level format the diskette. *mtools2* was posted in Volume 25, Issues 97-99. Volume 25, Issue 103 is a set of patches to *mtools* 2.0.5 from Henry van Cleef <vancleef@netcom.netcom.com> to support XENIX 286 systems. There were portability problems in the original release in regards to assuming that *ints* are at least 32 bits long. In XENIX 286, an *int* is 16 bits.

Recent patches appearing in *comp.sources.unix* include:

psroff had patches 5, 6, and 7 posted as Volume 25, Issues 32, 33, and 104. *psroff* allows both older C/A/T *troffs* and the newer *di-troffs* to work with Postscript printers and with HP Laserjet printers. Patch 5 is minor fixes mostly for compilation on 80286 style machines. Patch 6 is very important as several major features were broken and fixed by this patch. Patch 7 is fixes for *groff/di-troff* users using HP laserjets.


pathalias, the USENET map routing program, had patch 10 released by Peter Honeyman <honey@citi.umich.edu> as Volume 25, Issue 89. This is purely a bug fix patch and it is very small, but since so much depends on *pathalias*, I thought it worth mentioning.

No Reviews

The status postings show several projects in the re-review stage, but nothing appeared in *comp.sources.reviewed* over the past two months. Now that *comp.sources.unix* is back, this is not unexpected.

misc Tries an Experiment

Kent Landfield, the moderator of *comp.sources.misc* tried an experiment this time. He took a very large posting, made a *tar* file of it, and then compressed the *tar* file. He then uuencoded the compressed *tar* file and posted that. Even posted that way it was 42 parts (plus part 00 with the text on what the



Attack Memory Problems with Gusto

C-Heap by Library Technologies

ATTENTION C and Assembly Language Programmers! Here's the **ULTIMATE** in heap management libraries:

- Over 500 functions • 100% C-callable assembly language routines
- For Borland and Microsoft C and C++ compilers

Prevent, reduce or actively eliminate memory fragmentation. Use XMS/EMS/Virtual memory for data (no 64k limit). Up to 1600-fold speed improvement over malloc(). Up to 10-fold improvement in efficiency of memory usage.

- Lightning-fast disk I/O of linked list data • Exert total control over far heap and program block size • Use 2 near, 2 far heaps
- Implement a best fit malloc() in a flash • Use entire far heap as stack space • Efficient loading of data files • and much more!

C-Heap allows optimum utilization of your computer's memory under DOS. Extraordinarily complete documentation and many low-level functions let you play your heap like a violin. Many programs will experience a metamorphosis.

\$399 w/source \$199 w/o. No royalties. MC/VISA, purchase orders accepted. Call for free demo.

LIBRARY TECHNOLOGIES

800-767-4214 608-274-4224
P.O. BOX 56031-MADISON, WI 53705-9331

◆ Request 189 on Reader Service Card ◆

posting is and how to convert it back to a *tar* file). If posted as the normal *shar* format it would have been over 140 parts, probably the largest single posting ever tried. Reviews of this method were mixed, with a large amount of complaints on how it is harder to determine what is there, and to sort and deal with it, but I had no problems saving the files, running my concatenation script and feeding that to *uudecode* to convert the ASCII back to binary. It then uncompressed cleanly and produced a 7.5MB *tar* file.

And what was this grand experiment..., *pp* v6.0, a Mail Transfer Agent (MTA). *pp* is designed for high volume message switching, protocol conversion, and format conversion. *pp* supports X.400, RFC-822, and RFC-1148bis conversion between RFC-822 and X.400. *PP* is designed as a replacement for MMDF or sendmail. *pp* speaks X.400 (1984 and 1988), SMTP, JNT, UUCP, DECNET Mail-11, X.500, alias files, RFC-822 local delivery, and Fax Internetworking. No User Agents are provided, but a line oriented and an X-Window based management console program are provided. *pp* v6.0 was contributed by Steve Hardcastle-Kille <*S.kille@cs.ucl.ac.uk*> for Volume 27, Issues 24-66.

The shadow log-in suite for UNIX systems was rereleased by John F. Haugh II <*jfg@rpp386.cactus.org*> for Volume 26, Issues 54-64. New in release 3 is support for SVR4 style maintenance utilities and the grouping of the code into libraries to make maintenance easier. This suite provides shadow log-in/password management to many UNIX systems that do not yet have such support natively. One file was left out of the distribution and was posted as patch 1 in Volume 26, Issue 75.

Robert Davies <*robert@am.dsir.govt.nz*> contributed a new version of his C++ matrix package for Volume 26, Issues 87-91. *newmat04* supports matrix, upper and lower triangle, diagonal and symmetric matrices, row and column vectors and the element type float/double. Operators include *, +, -, inverse, transpose, submatrix, determinant, decomposition, triangularization, eigenvalues, sorting and fast Fourier transforms. It is intended for matrices in the size range 4x4 to 90x90.

Have an HP Laserjet with the Pacific Data Systems 25-in-One font cartridge? If so, Bill Walker's <*bkw@uecok.e-cok.edu*> *wroff* is what you need. It is a text formatter in the spirit of *nroff*, but designed specifically for this combination of hardware. *wroff* runs on UNIX, XENIX, MS-DOS and CPM-68K. It

does kerning and some other *troff* like items as well. *wroff* is Volume 26, Issues 97-101.

Ted Campbell <*tcamp@hercules.acpub.duke.edu*> contributed *sfs* the space flight simulator for IBM PC's with EGA or VGA, UNIX- PC's with MGR or UNIX with X11. A 21-part posting in Volume 27, Issues 1-21, *sfs* offers a graphics-based real-time animated simulation of orbital flight. You can simulate a complete range of orbital parameters and can also simulate multiple planets in a solar system. A particularly full map is given of the earth and can be displayed as viewed from the orbiting spacecraft, as a ground track map, or as a distance perspective in which the orbital track can be seen.

A compatible *regex* (regular expression) package without any licensing restrictions was contributed by Tatu Ylonen

SIMPLIFY APPLICATION DEVELOPMENT!

PINNACLE RELATIONAL ENGINE

Database Management Library for C and C++

- ✓ Fundamentally Relational Design
- ✓ Insanely Simple Programmer Interface
- ✓ C++ Class Header For Syntactic Elegance
- ✓ ANSI SQL Interface
- ✓ Windows DLL
- ✓ Liana™ WinApp Development Support
- ✓ Portable Server Architecture
- ✓ Same Engine Source for Multiple Platforms
- ✓ 90-Day Risk-Free Evaluation
- ✓ CALL 1-800-822-4437 NOW

Vermont
Database
Corporation

Vermont Database Corporation, 400 Upper Hollow Hill Rd, Stowe, VT 05672
1-802-253-4437 (voice) 1-802-253-4146 (FAX)

<ylo@ngs.fi> for Volume 27, Issue 23. It is fully compatible with the GNU *regex* library and can handle arbitrary data including binary patterns. It also can compile and run on 16-bit machines such as MS-DOS.

Those interested in genealogical research may want to get Steve Murphy's <murf@oakhill.sps.mot.com> *gcom* from Volume 21, Issues 72-78. *gcom* reads in GEDCOM format files containing genealogical data and merges them, utilizing not only name and date match heuristics, but familial ties as well.

In my February column, I mentioned Archie, the service that keeps track of which sites archive which data. Brendan Kehoe <brendan@cs.widener.edu> has updated his *archie* client in Volume 27, Issues 79-84. Version 1.3 of *archie* is his Prospero client for the *archie* service. Note, using this client requires TCP/IP access to an Archie server, which means you must be on an Internet.

- Last on the new release front is the latest release of *dmake*. Version 3.8 replaces version 3.7 and hopefully addresses all the little obscure bugs and features that remained. Dennis Vadura <dvadura@plg.waterloo.edu> has provided this version of the *make* utility. This package is the extended *make ala* BSD 4.4 release including many more features than the traditional versions. It includes support for UNIX, XENIX, MS-DOS, OS/2, and Atari-ST TOS. *dmake* 3.8 is Volume 27, Issues 101-142.

On the patch front, patch 9 was issued for *parseargs* in Volume 26, Issues 65 and 66 by Brad Appleton <brad@ssd.csd.harris.com>. *parseargs* provides a set of functions to parse command-line arguments. It can do much

more than the *getopt* variety of parser. Patch 9 is mostly bug fixes and was followed by patch 10 in Volume 26, Issue 116 for a bit more cleanup.

qbatch from Alan Saunders <tharr!alan> had its patch 2 posted for Volume 26, Issue 70 and patch 3 posted in Issue 85. Again, these were bug fix releases.

The KSH lookalike, *pdksh*, posted by Simon Gerraty <sjg@zen.void.oz.au> was updated to patch 1 in Volume 26, Issues 71 and 72. The build process was cleaned up and some portability issues were addresses.

The tin threaded newsreader had patches 6 and 7 posted by Iain Lea <stevax!lain> in Volume 26, Issues 76-82. New are support for Minix 386, more -M options for From lines, unread articles, and scrolling, plus some bug fixes. Patch 6 was in five parts and patch 7 in two parts.

PBMPLUS the multi-format image manipulation toolset was also updated to fix some bugs and to add several new programs. Jef Poskanzer <jef@well.sf.ca.us> provided patch 10 in Volume 26, Issues 106-110. New are *pgmcrater*, *ppmforge*, *ppmtoacad*, and *sldtoppm*.

Alternative Games

Two different versions of a "get the money game" called *sokoban* were contributed by two different authors. Kevin Solie <kevins@ms.uky.edu> contributed his *xsokoban* for Volume 13, Issues 1-2. This X-based game is a very incomplete implementation of an idea from a similar PC game.

The other version, *xsokoban2*, is from Joseph Traub <jt10+@andrew.cmu.edu> and was released in Volume 13, Issues 13-15. It provides a slightly different user interface than Kevin's version, but is essentially the same game.

Volume 13, Issue 3 provides *dr_mario* from Scott Noecker <noecker@seq.uncwil.edu>. This is a one-player lookalike version of Dr. Mario, a popular game for Nintendo that has nothing to do with the Mario Brothers series. It uses the standard keyboard and curses for the display driver.

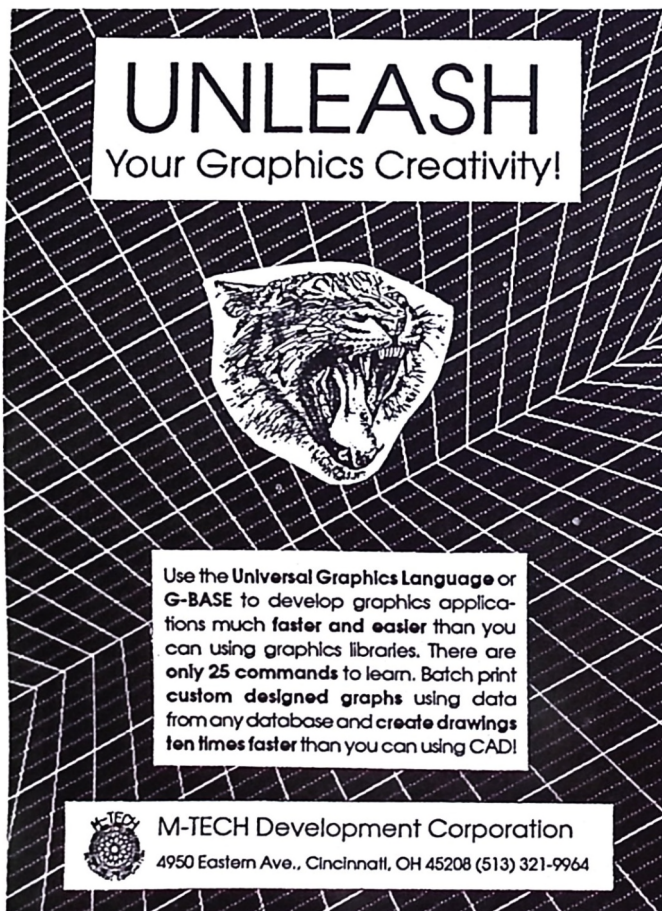
A complete revision of the WCST Nethack spoilers file updating it to version 7 was contributed for Volume 13, Issues 4-9 by Paul Waterman <wheaton!water>. A complete reformatting has been done to make the information easier to use, and of course, more changes and sections have been added.

A connect-five-in-a-row-game, *xmake5* was contributed by Chih-Hung Hsieh <hsiehch@spunky.cs.nyu.edu> for Volume 13, Issues 10-12. This game, written in C++ provides both a curses and an X-Window interface. The X version uses the athena widgets library.

For those with multi-user networks, a multi-player networked bridge game was contributed by Matthew Clegg <mclegg@cs.ucsd.edu>. *okbridge*, Volume 13, Issues 16-23, is a computer mediated bridge game that allows four players to participate in a game of rubber or duplicate bridge. The program handles dealing, scoring and communication of bids and plays. Issue 23 is a patch to fix a small problem in one of the distribution files.


Previews from alt.sources

Its been a quiet two months in *alt.sources*, only 4MB worth of notable postings (that or I am being more selective in what I consider notable). The most notable posting is a release of a *uucp* work-alike package from Ian Lance Taylor



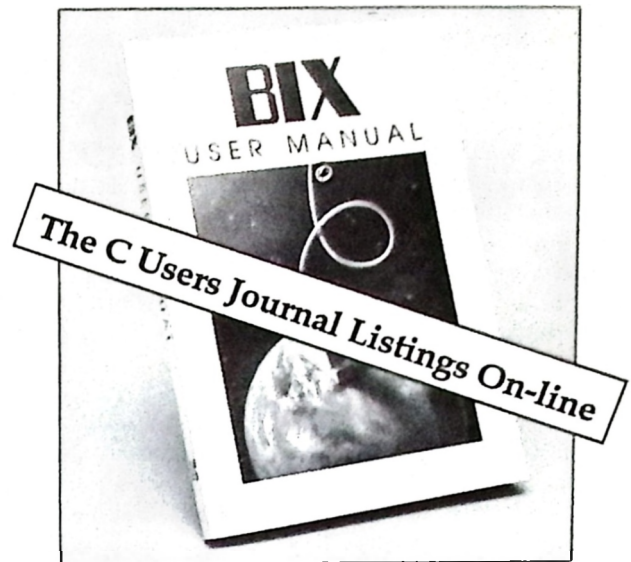
UNLEASH
Your Graphics Creativity!

Use the Universal Graphics Language or G-BASE to develop graphics applications much faster and easier than you can using graphics libraries. There are only 25 commands to learn. Batch print custom designed graphs using data from any database and create drawings ten times faster than you can using CAD!

 M-TECH Development Corporation
4950 Eastern Ave., Cincinnati, OH 45208 (513) 321-9964

◆ Request 467 on Reader Service Card ◆

Where Do You Go for Help When You're the Expert?



The Programmers Exchange on BIX!

Join any of these conferences in the programmers exchange:

<i>algorithms</i>	<i>Using algorithms to solve problems</i>
<i>ai.theory</i>	<i>Artificial intelligence and expert system theory</i>
<i>assembler</i>	<i>Assembly language</i>
<i>cad</i>	<i>Computer-aided design</i>
<i>c.language</i>	<i>The C programming language conference</i>
<i>c.plus.plus</i>	<i>Discuss the C++ programming language</i>
<i>editors</i>	<i>The Programming Editors Conference</i>
<i>games</i>	<i>Game programming and design</i>
<i>graphic.pgms</i>	<i>Programming and graphics</i>
<i>neural.nets</i>	<i>Neural Networks</i>
<i>ood</i>	<i>Object-oriented development conference</i>
<i>postscript</i>	<i>Postscript</i>
<i>soft.eng</i>	<i>Efficient and reliable software design</i>
<i>tech.notes</i>	<i>Collect and discuss useful programming code</i>
<i>unix</i>	<i>The Unix Conference</i>

All for just \$39 for three months plus \$3 per connect hour weeknights and weekends or \$6 per connect hour weekdays.*

With the programmers exchange you can:

- ◆ Get quick answers to tough coding questions.
- ◆ Interact with other C Developers.
- ◆ Download program listings for The C Users Journal.
- ◆ Download C Source code, utilities and other programs.
- ◆ Keep up with the latest hardware developments.
- ◆ Send and receive private e-mail with binary attachments.
- ◆ Chat with other C programmers in real time.

Subscribe to BIX!

Just set your telecommunications program for full duplex, 2400 or 1200 baud, 7 bits, even parity, 1 stop bit. Get your Visa, MasterCard, or American Express ready. Have your modem call your local BT Tymnet number. After connection is established, enter "A", then at "Please log in," enter "bix". When you are prompted for "Name?" enter "bix.cuj" and complete on-line registration. *You'll be on BIX and using the programmers exchange in just a couple of days!*

*Connect fees are for access via BT Tymnet and are subject to change. Call 800-336-0149 for your local access number. BIX handles billing for BT Tymnet connect fees. Other access available, call BIX at 800-227-2983 or 603-924-7681 for more information.

BIX



One Phoenix Mill Lane
Peterborough, NH 03458
1-800-227-2983 In NH (603) 924-7681

<ian@airs.com>. *taylor-uucp* v1.01 was posted on November 24, 1991 in 18 parts. It is a complete replacement for HDB style UUCP. It supports V2 style configuration (L.sys, L.devices) and BNU (aka HDB) style configuration files (Systems, Devices). It is a complete system, except for the fancy maintenance shell scripts and *uusched* (the latter is in the works).

Curt Mayer <hoptoad@curt> posted his disassembler for Z80/Z280 CP/M .COM files on November 27, 1991. The output is capable of being assembled and can detect code sequences by tracing jump targets.

Along with the CP/M disassembler, D'Arcy J. M. Cain <druid@darcy> posted on December 19, 1991 in three parts his Z80 CP/M emulator for UNIX. Most of the Z80 instruction set is implemented, except for the I/O section. The emulator also uses the UNIX shell commands to simulate some of the CP/M commands and uses the UNIX file system for drives. All I/O must go via the BDOS or BIOS as IN and OUT opcodes are not fully implemented.

On the opposite front, Quinn Jenson <jensenq@qcj.icon.com> posted a DSP56001 assembler on November 29, 1991 in four parts. The syntax was intended to be compatible with Motorola's syntax, but without the docs he could only guess. It does allow for UNIX based DSP code development for those not lucky enough to have a NeXT.

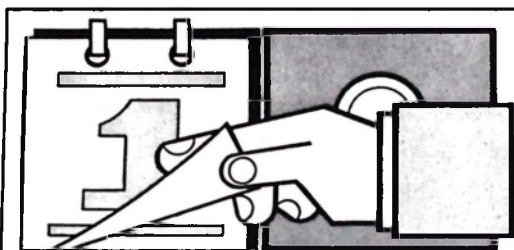
pcal, the postscript calendar program has been updated to Version 4.3 by Joseph Brownlee <jbr0@cbnews.cb.att.com> on December 16, 1991 in seven parts. *pcal* allows for creation

of personal calendars. New in 4.3 are generation of UNIX calendar files, move the previous and following month boxes around on the page, allow notes in any empty day box, change both note font and size via command line options, addition of the nearest keyword as in "workday nearest every 10th", plus about a page more worth of additions (something had to fill the 13000 lines of the posting).

Mayan Moudgill <moudgill@cs.cornell.edu> has posted his C++ socket library that adds UNIX and INET sockets to the *streams* functions. It was posted in late December, and then an updated posting was made on January 8, 1992 in two parts. Supported are convenient connection setup routines, use of the usual <& >> operators for i/o, support for out of band message transmission and reception, and an easy method of specifying per socket SIGIO, SIGURG and SIGPIPE handlers. Also include is an interface to the select system.

A gateway between the UNIX and the Citadel style BBS was posted on December 24, 1991 in four parts by Ken MacLeod <unidel@bitsko.slc.ut.us>. *uccico* implements the Citadel BBS networking for a UNIX system much like *uucico* does for UUCP. It handles USENET news and RFC822 mail conversion.

A large posting was *mixview*, posted by Robert Lau but written by Douglas Scott <doug@woof.columbia.edu> in 11 parts on January 12, 1992. It is an X-Window program that allows for editing sound files. It supports any BSD style system that has sound capabilities, such as Sun's and NeXT's. □



Calendar of Events

April

1-3 *Eleventh IEEE International Phoenix Conference on Computers and Communications*, Scottsdale, Arizona. Contact IEEE IPCCC-92, P.O. Box 8950, Scottsdale, AZ 85252, (602) 234-4477.

6-10 *LATIN '92*, International Symposium of Latin American Theoretical Informatics, São Paulo, Brazil. Sponsored by ACM. Contact Universidade de São Paulo, Instituto de Matemática e Estatísticas, Caixa Postal 20570; 55-11-813-9499.

27-30 *XWorld, The X Info Xchange*, New York. Sponsored by the X Journal. Contact SIGS Publications, 588 Broadway, Suite 604, New York, NY 10012, (212) 274-0646.

27-May 1 *USE inc. Spring Conference*, San Francisco. Contact USE, Inc., P.O. Box 461, Bladensburg, MD 20710, (301) 699-9336.

30-May 2 *Independent Computer Consultants Association 15th Annual National Conference*, St. Louis, MO. Contact ICCA at (800) GET-ICCA or (314) 997-4633.

May

5-7 *European Direct Marketing Conference*, Brussels. Contact European Direct Marketing Association, 34, rue du gouvernement provisoire, B-1000 Brussels, 32-2-217-63-09.

June

1-4 *Object Expo*, New York. Sponsored by SIGS Publications. Call (212) 274-9135.

8-12 *C Plus C++...in Action*, London. Sponsored by Boston University. Contact Boston University, Corporate Education Center, 72 Tyng Road, Tyngsboro, MA 01879, (508) 649-9731.

10-12 *International Conference on Intelligent Tutoring Systems*, Montreal. Sponsored by ACM. Contact University of Montreal, 2900 boul. Edouard-Montpetit, Dept I.R.O. Montreal, Quebec H3T 1J4, (514) 343-7019.

July

12-17 *AAAI-92*, San Jose, CA. Tenth National Conference on Artificial Intelligence. Contact AAAI-92, 445 Burgess Drive, Menlo Park, CA 94025, (415) 328-3123.

14-17 *Object Expo Europe*, London. Sponsored by SIGS Publications. Call (212) 274-9135.

20-24 *Logic at Tver '92*, Tver, USSR. Sponsored by ACM. Contact University of Tver, 33 Zhelyahova Street, Tver 170013 USSR.

August

10-13 *C++ Technical Conference*, Portland, OR. Fifth annual C++ conference sponsored by USENIX. Contact USENIX Conference Office, 22672 Lambert St., Ste. 613, El Toro, CA 92630.

24-27 *18th International Conference on Very Large Data Bases*, Vancouver, BC. Sponsored by ACM. Contact University of Alberta, 615 General Services Bldg., Edmonton, Alberta T6G 2H1 Canada, (403) 492-4589.

Calling Functions from Within a Function

Q I work with ANSI-C and I have a general problem. How is it possible to call one function with a variable number/types of arguments inside a second function with variable number/types of arguments, with the same arguments I called the second function?

For example, I have three functions with a variable number and variable type of arguments (*myfunction1*, *myfunction2*, *myfunction3* shown in Listing 1). I want to call the functions *myfunction1* and *myfunction2* from inside *myfunction3*, with the same arguments I called *myfunction3*. Is there an easy way to do that?

Thank you very much for your answer.

Willi Fleischer
Moerfelden, Germany

A The error involves the difference between a variable parameter list and a parameter list containing a value of type *va_list*. When you pass parameters to a function, the values of those parameters are pushed onto the stack. Usually the first or second parameter to a variable parameter function indicates directly (with a count) or indirectly (as with format specifiers) how many values have been passed.

The *printf* and *fprintf* functions expect to see the values on the stack. Each value has an address (its position on the stack). The *vfprintf* function expects its third parameter will be the address of the first value of a set of parameters on the stack. It then uses this address to retrieve those parameters. To use each parameter, it needs to know the type of the value. That information it gets from the format list specifiers, just like *printf* and *fprintf*. The type of the parameter is also used to find the next parameter.

Perhaps it might be instructive to look at typical definitions for these macros. Those in Listing 2 are from Microsoft C.

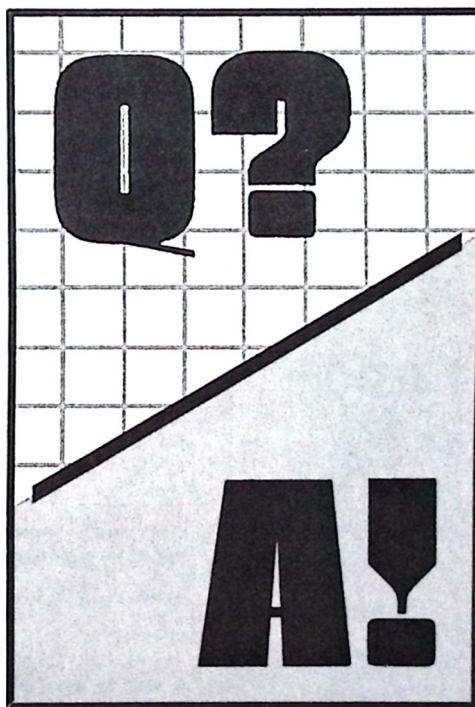
Variables of type *va_list* are actually addresses. *va_start* puts into the first argument the address of the first variable parameter. *va_arg* increments the first argument (*ap*) by the *sizeof* the type which is passed as the second argument (*t*), as well as yielding a value of that type.

How does *vfprintf* get to a value on the stack? Since it knows the type of argument (from the format list), it uses the *va_arg* macro with the appropriate type.

Your *my_function1* and *my_function2* require a list of values to be passed to them. When you used the *va_start* macro in *my_function3*, you retrieved the address of the first variable parameter that was passed to *my_function3*. You then passed that address to *my_function1* and *my_function2* and they produced garbage. You need to rewrite *my_function1* and *my_function2* such that they expect an address (i.e. type *va_list*). Listing 3 shows how they should look.

Q I've been programming in C for about two years now and have not seen anything in the literature about my question. The company I work for does quick and dirty production programming. We are in the process of moving from PL/I to C. The programming involves sequential

processing of mostly fixed fielded files. In PL/I we read the file into an input buffer and use the *string* function to load it into a structure. Listing 4 is an instance.



Kenneth Pugh, a principal in Pugh-Killeen Associates, teaches C language courses for corporations. He is the author of *C Language for Programmers and All On C*, and was a member on the ANSI C committee. He also does custom C programming for communications, graphics, image databases, and hypertext. His address is 4201 University Dr., Suite 102, Durham, NC 27707. You may fax questions for Ken to (919) 489-5239. Ken also receives email at kpugh@dukemvns.ac.duke.edu (Internet).

What this in effect does is load the structure *recin* from the input buffer *ltn*. There is no corresponding C function for loading such a structure and I've been trying to develop something that will do this. What I've come up with is Listing 5.

While the coding for *load_struct* works (at least in VAX C) I am not convinced this is the best way or even an effective way for doing what I desire. I realize the error checking is non-existent, but is it correct to assume that a structure of character arrays will have contiguous addresses? Is there another method used by more experienced programmers? I need something that I can put into a library that is very generic. How would you tackle this problem?

Tom Crosman
Brooklyn Park, MN

A On some machines individual fields in a structure do have packing bytes between them to align them to word boundaries, giving them contiguous addresses. Records which consist of character only (such as your example) tend not to have packing bytes.

Your method in general is fine. Since you asked for my solution, I've given it in Listing 6. Let me explain the modifications that I've made. The first is that one should usually never declare a variable in the same statement that declares the structure template. Anyone who wishes to use that template gets stuck with that extra variable. Second, I used an array of sizes for each of the fields. These could actually be picked up using the *sizeof* operator. As another alternative, one could

#define the size of each field and use those in the initialization list.

I prefer using an array for the sizes instead of passing them in the parameter list. The declaration can be close to the structure template. Any changes in order or size of the fields can be simply coordinated. Using an array to pass the sizes also

Listing 1

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>

void myfunction1(char *format, ...)
{
    va_list arg_ptr;
    va_start(arg_ptr, format);
    vfprintf(stdout, format, arg_ptr);
    va_end(arg_ptr);
}

void myfunction2(char *format, ...)
{
    FILE *fp;
    va_list arg_ptr;
    fp = fopen("TEST.DAT", "a+");
    va_start(arg_ptr, format);
    vfprintf(fp, format, arg_ptr);
    va_end(arg_ptr);
    fclose(fp);
}

void myfunction3(char *format, ...)
{
    va_list arg_ptr1;
    va_list arg_ptr2;
    va_start(arg_ptr1, format);

    myfunction1(format, arg_ptr1);

    /* here I want to use the arguments of myfunction3(),
    va_end(arg_ptr1); but this code does not work */

    va_start(arg_ptr2, format);
    myfunction2(format, arg_ptr2);

    /* here I want to use the arguments of myfunction3(),
    va_end(arg_ptr2); but this code does not work */
}

int main()
{
    char msg[] = "message";
    myfunction1("\n%s=%d=%f", msg, 2, 3.0); /* this works
fine */
    myfunction2("\n%s=%d=%f", msg, 2, 3.0); /* this works
fine */

    /* the following call of myfunction3() does not work.
    I want to have the same result, as
    if I call myfunction1() and myfunction2() isolated */

    myfunction3("\n%s=%d=%f", msg, 2, 3.0);
}

/* End of File */
```

Listing 2

```
typedef void *va_list;
#define va_start(ap, v) ap = (va_list)&v + sizeof(v)
#define va_arg(ap, t) ((t*)(ap += sizeof(t)))[-1]
#define va_end(ap) ap = NULL
/* End of File */
```

Announcing version 2:

Victor Image Processing Library Use Victor to develop powerful image applications

Work with images of any size -- use conventional, expanded, and extended memory

Now your applications can support 8-bit color and gray scale images of any size because Victor gives you complete control over conventional, expanded, and extended memory.

Display on Super VGA

Display images on EGA/VGA and super VGA up to 1024 x 768 256 colors.

Load & save PCX/TIFF/GIF/BIN

Handle images from any source, or create translation programs between the popular file formats.

Gray scale and color images

Powerful image processing for all images -- your software can have features like: zoom, resize, brighten, contrast, sharpen, outline, linearize, matrix conv, colorize, & more.

ScanJet and LaserJet support

You can offer device control for gray scale scanning -- AND print halftones at any size.

Victor supports Microsoft C, QuickC, and TurboC, includes demonstration and prototyping software, and full documentation. Source code available.

Victor Library version 2, \$195

Call (314) 962-7833 to order VISA/MC/COD

Catenary Systems 470 Belleview St Louis MO 63119 (314) 962-7833



Image-based applications can be developed in MSC, QuickC, and Turbo C environments.



Your application will be able to print halftones at any size.



Give your applications support for scanner and laser printer.

◆ Request 253 on Reader Service Card ◆

Listing 3

```
void myfunction1_a(char *format, va_list arg_ptr)
{
    vfprintf(stdout,format,arg_ptr);
}

void myfunction2_a(char *format, va_list arg_ptr)
{
    FILE *fp;

    fp = fopen("TEST.DAT","a+");
    vfprintf(fp,format,arg_ptr);
    fclose(fp);
}

void myfunction3(char *format, ...)
{
    va_list arg_ptr1;
    va_list arg_ptr2;

    va_start(arg_ptr1,format);
    myfunction1_a(format,arg_ptr1);
    myfunction2_a(format,arg_ptr1);
    va_end(arg_ptr1);
}

/* End of File */
```

Listing 4

```
DCL 1 RECIN,
  2 NAME    CHAR(30),
  2 ADDRESS CHAR(20),
  2 CITY    CHAR(15),
  2 STATE   CHAR(2),
  2 ZIP     CHAR(5);

...

READ FILE (FILIN) INTO (INN);
STRING(RECIN) = INN;

...

/* End of File */
```

Listing 5

```
#define MAX 1000

void load_struct(instruc, instr, ...);

/***** main function *****/

load_struct(instruc, instr, va_alist)
char *instruc;
char *instr;
{
    int k=0;
    int strnglen;
    va_list ap;
    va_start(ap);

    while((strnglen = va_arg(ap,int)) != NULL)
    {
        for(k = 0; k < strnglen-1; k++)
            *instruc++ = *instr++;

        *instruc++ = 0;
    }
    va_end(ap);
}

/* End of File */
```

Call for Papers

The C Users Journal is seeking articles on the topics below. If you have an idea for a related story, or experience that would qualify you especially to write on one of these topics, contact *The C Users Journal* editorial staff for *Author Guidelines* at:

The C Users Journal
Attn: Managing Editor
1601 W. 23rd St., Suite 200
Lawrence, KS 66046-2743
(913) 841-1631

We prefer practical treatments of real programming problems, or tutorials that make theory or complex issues more accessible to practicing programmers. Our emphasis is on C and C++ or on tools used in conjunction with these languages.

We pay at rates that are competitive with other national technical journals. We believe that our editorial assistance is better than most. You don't have to be a professional writer to meet our acceptance criteria, but you must have something to say. Again, contact the editorial staff for *Author Guidelines*.

Graphics

Proposals due 4/13/92; manuscripts due 5/11/92

Suggested topics: Writing C or C++ code that ports easily among the major GUI environments; useful graphic classes and libraries; algorithms for drawing, shading, windowing, scaling, etc. that are efficient and/or elegant; techniques for visualizing data; manipulating gray-scale and color images.

Overworked topics (require fresh perspective): graphic object classes, graphic file reformatters.

Communications and Networks

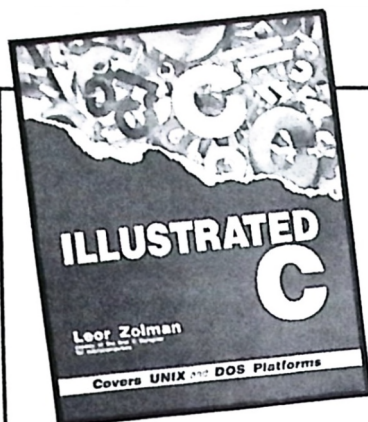
Proposals due 5/4/92; manuscripts due 6/15/92

Suggested topics: Interfacing C or C++ code to any of the popular commercial network packages; strategies for implementing simple but robust network protocols; programming popular ports or multiplexors in C or C++; tutorials on common interface protocols, interchange formats; user reports on major communications, interchange packages; simple handshake algorithms; real-life applications of queuing theory.

Overworked topics (require fresh perspective): UART drivers, kermit protocol.

**Build
better
applications
now with. . .**

**JUST
RELEASED**



Illustrated C

by

Leor Zolman

CWJ columnist and author of BDS C

Discover the WHY and HOW of application design and development in C. Explore the construction of several different applications from start to finish. Chapter after chapter you'll develop your skills through in-depth tutorial and detailed code.

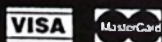
Only \$29⁹⁵

(\$39⁹⁵ with disk)

CALL TODAY!

913-841-1631

FAX 913-841-2624



RID
publications, inc.

simplifies the function, since it is no longer concerned with a variable parameter list.

To show an additional use for the array, I included a print function. It requires most of the same parameters as the load function.

To make the function more generic, I added a *nul_terminated* flag. Some programmers do not like using the extra character space to hold a terminator in each field. The *nul*, if present, can signify the end of a value less than the field length. If not present, the field length is the size of the field. Though this requires a slight bit more coding, it does save significant disk space if you store thousands of copies of a structure.

If you were concerned with the packing of the fields in either the input record or the output structure, then you could add an array of field addresses to the calls. If that is necessary, I might suggest not using a generic function and simply hard coding any record conver-

sions required. At some point the work of providing and using a generic interface exceeds the benefit.

Reader Feedback

Coding style

I notice in your *find_maximum* function that you declare temporary storage for the return value. Is there some reason this is preferred over just returning the value directly? This allows the simplifications in Listing 7.

Also in your listings, for the function *put_line*, it seems to me that *putchar* would be preferred to *printf("%c",...)*. It would not have to process the format string and convert the character before putting it on *stdout*.

I enjoy your column and comments.

Edward C. Sarlls, III
Houston, TX

Listing 6

```
struct rec {
    char name[31];
    char address[21];
    char city[16];
    char state[3];
    char zip[6];
};
static int field_sizes[] = {31, 21, 16, 3, 6};
/*
Alternatively, this could be written as:

static int field_sizes[] = {    sizeof(rec.name),
    sizeof(rec.address),
    ...
};
*/
#define field_count (sizeof(field_sizes)/sizeof(int));

main()
{
    char record_in[MAX];
    struct rec record;
    ...
    while(fgets(record_inn, MAX, filin) != NULL)
    {
        load_struct(&record, record_in, field_sizes,
field_count, TRUE);
        print_struct(&record, field_sizes, field_count, TRUE);
    }
}

print_struct(record, field_sizes, field_count, nul_terminated)
char *record;          /* Record to print */
int field_sizes[];      /* Size of fields */
int field_count;        /* Number of fields */
int nul_terminated;     /* If fields are nul terminated */
{
    int field;
    int length;
    char *pc;
```


Listing 6 — Cont'd

```
printf("\n");
pc = record;
for (field = 0; field < field_count; field++)
{
    if (nul_terminated)
        printf("%s:", pc);
    else
    {
        length = field_sizes[field];
        printf("%.s:", length, pc);
    }
    pc += field_sizes[field];
}

load_struct(record, record_in, field_sizes, field_count,
nul_terminated)
char *record;
char *record_in;
int field_sizes[];
int field_count;
int nul_terminated; /* If fields out should be nul terminated */
{
    int field;
    int length;
    char *pc;
    char *pc_in;

    pc = record;
    pc_in = record_in;
    for (field = 0; field < field_count; field++)
    {
        length = field_sizes[field];
        strncpy(pc, pc_in, length);
        if (nul_terminated)
        {
            pc[length-1] = '\0';
            pc_in += length - 1;
        }
        else
            pc_in += length;
        pc += field_sizes[field];
    }
}

/* End of File */
```

Listing 7

```
int find_maximum(one, two, three)
int one, two, three;
{
    if (one > two)
        if (one > three)
            return one;
        else
            return three;
    if (two > three)
        return two;
    else
        return three;
}

... or ...

int find_maximum(one, two, three)
int one, two, three; /* This could be a macro if you're
                      careful about side effects. */
return ((one>two)?((one>three)?one:three):((two>three)?two:three));

/* End of File */
```



If you need to create numerical solutions beyond 640K on DOS and unleash 386 or 486 power, we have some real good news for you.

CSL32 is a brand new version of CSL for 32-bit, extended-DOS or window programming, with the Watcom C 386 or Intel C 386/486 compiler. Call Now! Find out about our *CSL32 Introductory Offer*.

Using CSL32 with the Intel or Watcom compiler, you may now build programs, in a flat memory model up to 4 gigabytes, which include numerical solutions for extended DOS or Microsoft Windows.

CSL is a distinguished C library for numerical solutions.

- Over 500 functions
- Mature & Reliable
- Source Code Option
- Guiding Examples
- Application Notes
- Validated Code
- Easy to Use
- Maintenance Option
- Runtime Distribution Rights
- CSL32 is compatible with CSL

CSL versions are also available for DOS, SCO UNIX V/386, XENIX, ESIX, HP/Apollo and SunOS. Call Eigenware to obtain CSL parts list and pricing information.

CSL, CSL32 and Eigenware are registered trademarks of Eigenware Technologies. Other brand and product names are trademarks or registered trademarks of their respective holders.

CSL Eigenware Technologies

13090 La Vista Drive
Saratoga, CA 95070
Phone: 408-867-1184

◆ Request 107 on Reader Service Card ◆

Listing 8

```
int find_maximum(one, two, three)
int one, two, three;
{
    int ret;
    if (one > two)
        if (one > three)
        {
            ret = one;
            goto end;
        }
    else
    {
        ret = three;
        goto end;
    }
    ...
end:
    printf("find_maximum returning %d", ret);
    return ret;
}
/* End of File */
```

I tend to use an automatic variable for the return value from a function. That makes it easier to put a *printf* statement in the code to print the return value of the function. Or if you are a debugger person, it makes it easy to watch the value.

The disadvantage is a slight decrease in speed. If I made it a register variable (or if the compiler does so automatically), even that should not be a problem.

Listing 9

```
if (one > three)
{
    printf("Find maximum returning %d", one);
    return one;
}

/* End of File */
```

Listing 10

```
void get_low_high(int a, int b, int c, int *low, int *high)
{
    *low = (a < b) ? a : b;
    if (c < *low) *low = c;
    *high = (a > b) ? a : b;
    if (c > *high) *high = c;
}

/* End of File */
```

If you decide to change the return value of the function that uses an expression, and then with the local parameter, you only have to change the expression in one place. If you have debugging output, you need to change it in two.

I must admit that I have had this style for a long long time. In one of my C classes that I taught back in the early '80s, I had a student who insisted that parentheses were required around the expression that follows the *return* statement. It turns out that all the examples of *return* statements he had seen had complex expressions around them (as the one in your second example). Psychologically the parenthesis were needed to surround the expression and "make it one." That points up the other coding style that I use quite often (see Listing 8). I state in *All on C* that having a single *return* statement with *gotos* is preferable to having multiple *return* statements. If I want to trace the *return value*, with multiple *return* statements, I have to do something like the code shown in Listing 9.

If I were using a debugger, there would be several breakpoints to set (assuming the function was long enough that I simply didn't single step through it).

The difference in the complexity of code between multiple *returns* and multiple *gotos* (to the end of a function) does not seem to be a big issue, at least to me.

I may get hundreds of letters regarding this seemingly idiosyncratic style of programming (or maybe with an adjective using only the first two syllables). Before that occurs, I wish to make a few caveats regarding it. First, I try to program the logic not to require multiple *returns/gotos*. Hence the original listing has neither in it. Second, there should be a single label at the end of the function with a standard name (say *end*), that is the label for the *goto*. If that is the case, then *ret = xxx; goto end* takes on the same meaning as *return xxx;* (KP)

Q I just read your question and answer article entitled "Using *typedef*" in the December 1991 issue of *The C Users Journal*. I am writing in reference to the question regarding the writing of a function returning the lowest and highest integer out of the three integers passed. Listing 10 is my attempt at answering this question.

New Windows DLLs

CrystalCOMM/dBC III PLUS for Windows

Now you can write your data base and modem communication applications under Windows. These Windows libraries support Microsoft 6.0 and Borland C++ 2.0, and are also available in DOS versions. The libraries support both dll(dynamic linked library) and standard library interfaces under Windows. The libraries fully support the Windows application environment rules for building friendly applications. Products include library object, source(optional), Windows example programs using the library, and comprehensive documentation.

CrystalComm for Windows \$175 (source version \$350)

CrystalComm is a communications library written in C for the Windows environment that supports the development of modem or serial port communication programs for the PC. It includes all of the necessary functions to maintain consistency in the Windows environment. CrystalComm supports XMODEM, YMODEM, KERMIT, and ASCII communication protocols through a simple, structured, and flexible function interface. It is also available for DOS.

dBc III PLUS for Windows \$450 (source version \$650)

dBc III Plus for Windows is a database library written in C that provides full data compatibility with dBASE III. The library enables you to build applications and run them independently or in concert with dBASE. dBC III PLUS for Windows supports locking so that your applications can be used in a multiuser or networked environment. You can create, access, and update database files for your Windows application. dBC III PLUS for Windows provides you with the speed, standardization, and custom capabilities not available in other libraries.



Crystal Software

P. O. Box 43, Amasa, MI 49903, USA - PAX (906) 822-7994x99
(906) 822-7994

Listing 11

```
#include <stdarg.h>
int maximum(int count, int first,...);
main()
{
    int ret;
    ret = maximum(6, 2,3,4,5,9,8);
    printf("Maximum is %d\n", ret);
}

int maximum(int count,int first,...)
{
    va_list arguments;
    int i;
    int value;
    int maximum = first;
    va_start(arguments, first);
    for (i = 0; i < count - 1; i++)
    {
        value = va_arg(arguments, int);
        if (value > maximum)
            maximum = value;
    }
    va_end(arguments);
    return maximum;
}
/* End of File */
```

With all due respect, I think this approach is more eloquent and less convoluted than the methods offered in Listing 2 (December 1991 issue) and Listing 3 on page 120 (December 1991 issue) and in Listing 4 on page 122

Listing 12

```
void find_min_max(one, two, three, minimum, maximum)
int one, two, three;
int *minimum;
int *maximum;
{
    if (one > two)
        *minimum = two;
    else
        *minimum = one;
    if (three < *minimum)
        *minimum = three;

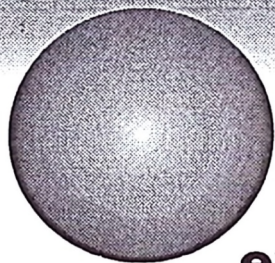
    if (one < two)
        *maximum = two;
    else
        *maximum = one;
    if (three > *maximum)
        *maximum = three;

    return;
}
/* End of File */
```

(December 1991 issue), whether or not you choose to use the conditional shorthand.

I am also puzzled by the fact that you were a member of the ANSI C committee and you didn't use prototypes in your code. Did you have a special reason for not prototyping your examples?

S.J. Stern
Bothell, WA



**We care
about the
environment**

**That's why we print
with SOY INK.**

The **G** Users Journal™



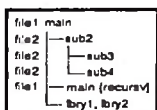
**PRINTED WITH
SOY INK**

Trademark of American Soybean Association

C AND C++ DOCUMENTATION

NEW! C-METRIC™ (\$49) COMPLEXITY / QUALITY

- Calculates "Cyclomatic" path complexity for functions and system
- Calculates lines with Comments, Code, and 'C' Statements

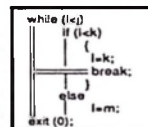


C-CALL™ (\$59) FUNCTION HIERARCHY

- Tree-Diagram showing function hierarchy
- Table-Of-Contents of functions versus files
- Detailed cross-reference of functions

C-CMT™ (\$59) FUNCTION COMMENT

- Generate/Insert function comment blocks
- Re-run anytime to update comment blocks
- Retains any user-generated comments



C-LIST™ (\$49) LIST OR REFORMATS

- Action-Diagrams show logic/control flow
- Reformats to various standardized formats
- Line numbers, page numbers, and titles

C-REF™ (\$49) CROSS-REFERENCES IDENTIFIERS

- Summary list of Local/Parameter/Global/Define Identifiers used
- Detailed Cross-Reference of Identifiers vs Function/Line usage
- Produces Class-Hierarchy Tree-Diagram for C++ Classes

SPECIAL OFFER: (\$189) Complete C-DOC™ Package

- All 5 programs fully integrated as 1 overall C-DOC program (DOS)
- NEW! special offer includes OS/2 protected-mode program
- Unconditional 30-DAY Money-Back Guarantee of Satisfaction !!

ACCURATE, and Fully AUTOMATIC (no source changes needed)

SOFTWARE BLACKSMITHS INC

6064 St Ives Way, Mississauga
ONT Canada L5N-4M1

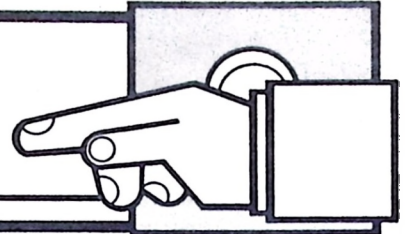


(416) 858-4466

Call for Immediate Shipment

◆ Request 183 on Reader Service Card ◆

Advertiser Index



Advertiser.....	Reader Service #	Pg. #	Advertiser.....	Reader Service #	Pg. #	Advertiser.....	Reader Service #	Pg. #
A.T. Barrett & Assoc.	380	.. 47	FFE Software Inc.	149	.. 113	Pathfinder Associates.....	165 55
AT&T	150	... 23	Faircom	128	... C4	Pauwels & Partners Software.....	209	.. 123
AT&T	**	... 25	Frontline Test Equipment.....	**	.. 105	Perennial.....	108 50
AT&T	**	... 27	Gimpel Software	**	... 83	Quadbase Systems Inc.	277 53
AT&T	**	... 29	Gotoless Conversion.....	**	... 76	Qualware.....	382	.. 124
Accelerated Technology Inc.	203	... 16	Greenleaf Software, Inc.	240	... C2	Quantum Software Systems, Ltd.	478 19
AdamSoft	146	.. 127	Guideware Corp.	132	... 71	Quarterdeck Office Systems.....	201 13
Advanced Design Solutions.....	142	... 75	HPI.....	118	... 66	Quinn Communications	154 47
AnSoft.....	412	.. 125	Hyperlogix.....	161	... 98	Quinn-Curtis.....	129 38
Ancier Technologies.....	184	.. 124	ITI Logiciel.....	**	... 93	Raima Corporation	296 9
At Last! Software Inc.	187	... 28	Illustrated C.....	495	... 88	Realtime Control, Inc.....	388	.. 128
Austin Code Works.....	151	... 2	Illustrated C.....	495	.. 108	Rogue Wave	124 61
Az-Tech Software Inc.....	245	.. 123	Inductive Logic	137	... 77	Roundhill Computer Systems.....	**	... 73
BIX.....	205	.. 103	Island Systems.....	143	... 12	Ryle Design.....	110	.. 127
Barton Creek Software	471	.. 123	Island Systems.....	144	.. 128	SET Labs.....	210	.. 116
Base Technology.....	204	... 18	KADAK Products, Ltd.	451	... 52	SeaBreeze Software	**	... 31
Black Ice Software, Inc.....	300	... 77	Kandu, Inc.....	481	... 69	Sequiter Software Inc.	166	... C3
Blaise Computing, Inc.....	102	... 5	Killdeer Software.....	117	.. 125	Siener Soft, Inc.	479 88
Borland International.....	465	... 11	Knowledge Dynamics Corp.	453	... 57	SofSolutions.....	200 14
Burton Systems Software	430	.. 125	LNB Software.....	198	... 96	SoftC Ltd.....	337 36
Byte Designs, Ltd.	111	.. 117	Legend Communications.....	192	.. 127	Softran.....	498 97
Byte-BOS Integrated Systems ..	136	... 40	Library Technologies.....	189	.. 100	The Software Annex	109 63
C::napse.....	**	... 127	M-TECH Development Corp.	467	.. 102	Software Blacksmiths	173	.. 127
C Users Bookstore	**	... 81	MIX Software.....	120	... 68	Software Blacksmiths	183	.. 111
C Users Group.....	**	... 82	MMC AD Systems	152	.. 123	Software Environments Ltd.....	371	.. 125
C Users Journal.....	**	... 79	Mark Williams Co.	**	... 1	Software Ingenuities.....	170 20
Catenary Systems.....	253	.. 106	Marian Software.....	145	.. 125	Softway Inc.	186 48
Cater Software	491	... 39	Mayer & Bunge Informatica, Ltd.	104	.. 125	Star Guidance Consulting, Inc.....	163	.. 126
Certified Scientific Software.....	208	... 124	Meridian Technology Corp.....	**	.. 125	Strategic Software Design Inc.....	387 85
Chirp Technical Services	**	.. 124	Micro Digital Inc.....	202	... 86	StratosWare.....	414 7
ChoiceWare	362	... 63	Microsoft	131	... 37	Strom Systems, Inc.....	447 92
Cobalt Blue.....	105	... 74	Microsoft	**	43,45	Stylus Software Pty Ltd.	196	.. 128
Code Farms, Inc.	127	... 56	Network Dynamics.....	419	.. 128	The Symmetry Group.....	177 70
Computer Solutions NW.....	445	... 65	Network Integrated Services.....	456	... 42	T & T Computer Products.....	247 87
Copia International Ltd.....	**	... 73	Network Research	494	.. 124	TauMetric	122 84
Coronado Enterprises	273	.. 127	Networks	464	.. 126	The Trachtman Group.....	125	.. 127
Cosyfor.....	175	... 42	Neuro Sym Corp.	327	.. 123	Trio Systems	169 26
Creative Programming	123	... 54	Nu-Mega Technologies.....	141	... 3	U.S. Software Corporation	476 22
Crystal Software, Inc.	**	.. 110	OPUS Software.....	159	... 72	V Communications	179 91
Decision System Software	112	... 78	On Time Marketing.....	101	... 87	Vault Corporation.....	162 44
Disk Software, Inc.....	116	... 93	Gary R. Olhoeft.....	475	.. 126	Vermont Database Corp.....	**	.. 101
Drumlin.....	363	... 17	Opt-Tech Data Processing.....	199	.. 126	WATCOM Products Inc.	126 15
Dyad Software Corp.	459	... 49	Optimate Systems	188	.. 123	WISE Software	364	.. 127
EMS	119	.. 126	PCX.....	130	... 10	Wenham Software Co.	138	.. 127
Eigenware Technologies.....	107	.. 109	PDQ Software	100	.. 125	Willies' Computer SW Co.	322 85
Eighteen Eight Laboratories	121	.. 128	PKWare Inc.	185	... 35	Wilsoft, Inc.	115	.. 123
EmmaSoft.....	197	.. 123	PSW/Power SoftWare.....	133	.. 125	Zircel Software.....	433 51
			Paladin Software, Inc.	106	... 18			

*This index is provided as a service to our readers.
The publisher assumes no liability for errors or omissions.*

****This advertiser prefers to be contacted directly.**

Listing 13

```
typedef double SPEED;
typedef double DISTANCE;
typedef double TIME;

SPEED low_speed, high_speed;
DISTANCE short_distance, long_distance;
TIME brief_time, long_time;
```

/* End of File */

Listing 14

```
SPEED compute_speed(DISTANCE distance, TIME time);
```

/* End of File */

A Eloquence is in the eye of the beholder. Some people might consider the second part of Listing 7 (from Mr. Sarlls letter previously in this column) the most eloquent. I don't particularly prefer any of my Listings 2, 3, or 4. My favorite is Listing 5 (from the December 1991 issue, reproduced here as Listing 11). It computes the maximum for any number of input parameters. The logic in your sample matches the logic in that listing.

I guess I could have arranged the logic in my function as shown in Listing 12. It matches your logic and has fewer lines than my previous listing. Unless I am going to call a routine a few thousand times, I tend to stick with whatever I come up with first that works. Also, I usually avoid using the conditional expression operator, for the reasons explained in last month's column.

As far as prototypes, I usually do not include them unless they are required by ANSI C or they are essential to the answer. The information contained in prototypes is mostly redundant. The case of the variable parameter function (as in Listing 10) requires one. When I do need prototypes (e.g. for C++), I let the compiler or PC-Lint generate a file of them.

Thank you for your feedback. (KP)

typedefs and lint

In my column a few months ago, I answered a question regarding *typedefs*. At the recent C-Forum sponsored by the Wang Institute of Boston University, I bumped into Jim Gimpel, the author of PC-Lint. He told me that the latest version of PC-Lint has an option for strong typing. This means that it can report errors in the use of *typedefed* variables which the compiler would just ignore. For example, given the code in Listing 13, the assignment of

```
short_distance = high_speed;
```

is accepted by the compiler without question since both variables are declared to be type *double*, once the *typedefs* are resolved into the underlying types. However PC-Lint can yield an error message, if strong typing is turned on.

As another example, a function declared as shown in Listing 14 will give a PC-Lint error if it is passed the code in Listing 15.

There are many options available for strong typing, which are all described in the manual. In addition, you can declare that particular arrays can only be indexed by variables of par-

Listing 15

```
low_speed = compute_speed(long_time, short_distance);

/* End of File */
```

Listing 16

```
/* lint -index(c, INDEX, HISTOGRAM) */

typedef unsigned int INDEX;
typedef int HISTOGRAM
#define SIZE (INDEX) 10

HISTOGRAM my_array[SIZE];
INDEX good_index;
int not_good_index;

my_array[good_index] = (HISTOGRAM) 5;
my_array[not_good_index] = (HISTOGRAM) 7;

/* End of File */
```

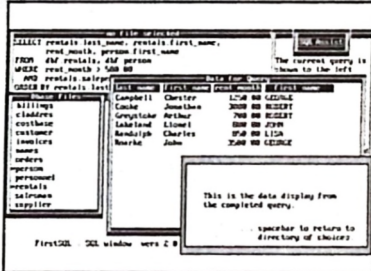
ticular types. For example, arrays of type *HISTOGRAM* can only be indexed by variables of type *INDEX*. This would look something like Listing 16.

The *not_good_index* reference to *my_array* would yield an output error.

For those who are considering changing to C++ purely for its type-checking abilities, I suggest you look at PC-Lint as an alternative. (KP) □

FirstSQL™

First in Innovation



FirstSQL SQL window - ver 2.0

"[FirstSQL] understands the nature of SQL better than any other product I tested"

Joe Celko
DBP&D 7-91

Complete Relational DBMS


- ☐ ANSI & DB2 Compatible SQL
- ☐ Direct Access to XBASE files
- ☐ The only total referential integrity
- ☐ Single User or Network

4GL Application Development Language

- ☐ Building Applications made simple
- ☐ Full windowing facilities
- ☐ Reporting Capabilities (with fonts)

ALSO AVAILABLE—FirstSQL C EMBEDDED SQL for C
Utilize the power of SQL in your C Applications

C++ Support, Too!



PCXT/AT&Compatibles · 512K · MS-DOS (2.1 up)

Phone: (510) 232-6800 Fax: (510) 237-7433

FFE Software, Inc. · P.O. Box 1519 · El Cerrito, CA 94530

◆ Request 149 on Header Service Card ◆

Gadgets

Update

CUG297 Small Prolog

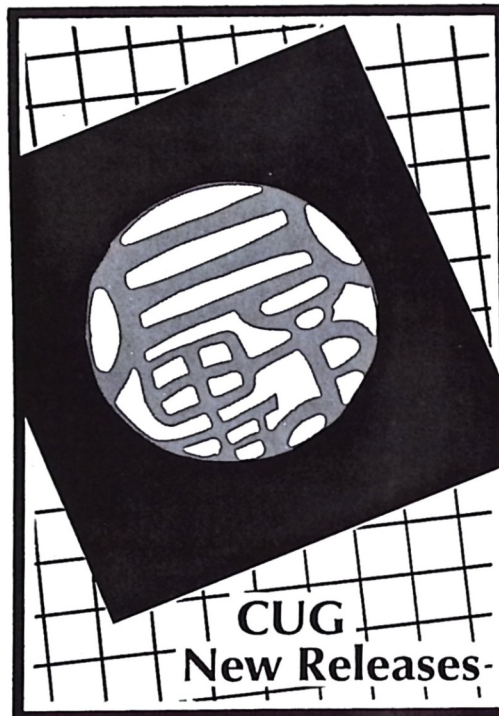
Henri de Feraudy (FRANCE) has released v2.0 of his Small Prolog. The new version offers much better debugging facilities, and a 32-bit executable compiled by GCC-386 (CUG359).

CUG327 Panels for C

J. Brown (KS) has released v2.3 update of his shareware window package, Panels for C. This version includes these new features: OS/2 support, Turbo C support, utilizing the PATH environment variables to find panel definition files, allowing the inclusion of panel definitions in the C source program, Interactive Panel Design (IPD) utility.

CUG351 UltraWin

Kevin L. Huck (MO) has released v2.10 of his shareware, Ultrawin. This new version includes new features: unlimited overlapping windows, background printing, PC timer control, mouse and graphic support, and enhanced data entry capabilities. Also included are a hypertext help engine and and EGA/VGA font editor. Also released is InTUItion 1.10, a textual user-interface library that includes an interface construction program that al-



lows UltraWin users to interactively create dialog boxes, menus, pick lists, forms and more using a mouse. Source code can be automatically generated to perform processing on each item, saving hours of tedious hand coding and debugging.

New Releases

CUG361 Gadgets and Term

Jack E. Ekwall has contributed a function library Gadgets, a group of UNIX-like tools for DOS; and Term, a collection of computer buzz-words. Gadgets provides functions such as popup/dropdown window, drawing box, screen and cursor manipulation, keyboard input, color, date, printer and mouse control, and file manipulation. Some of the functions are lifted from CUG273 Turbo C Utilities. The library is linkable to Turbo C v2.0. These UNIX-like tools offer a solution to the DOS command line interface pipeline problem. Term includes 634 topics and 32 historical notes/observations about computer buzz-words. This text is in a text-indexed sequential form which can be read by a display program, VU. The distribution disk includes source code for the library and documentation.

Kenji Hino is a member of The C Users' Group technical staff. He holds a B.S.C.S. from McPherson College and an undergraduate degree in metallurgy from a Japanese university. He enjoys playing drums in a reggae band.

Zinc Interface Library

Comments by David Brumbaugh

Introduction

The Zinc Interface Library is a C++ user interface library from Zinc Software Inc. for PC compatible computers. It supports MS-DOS text mode, MS-DOS graphics mode and MS-Windows 3.x interfaces. I'm reporting on Zinc Version 2.0 for Borland C++. Zinc also supports the Zortech C++ compiler.

C programmers moving to C++ and experienced C++ programmers will find the Zinc Interface Library helpful. I've been using it for over a year on projects I've been working on at home.

Features

Zinc's primary feature is a C++ class library. It consists of class definitions, object code and optionally, source code for those classes. The class library is designed so that your application only needs to have one set of source code for writing applications in text mode for MS-DOS, graphics mode for MS-DOS (CGA, EGA, VGA, Hercules compatible) and MS-Windows 3.x.

The user interface created in all three modes is SAA compliant. This means it has windows, menus, dialogue boxes, list boxes, optional mouse support, and all the other features users have come to expect in modern software.

All three modes, text, graphics and MS-Windows have the same basic look and feel. The `UI_DISPLAY` class encapsulates the three display types in its descendants. The `UI_DISPLAY` class defines all the things that a program can do to a user's display.

Text mode applications use the PC extended ASCII graphics set for windows. The programmer has several options when using text mode:

1. The application can automatically detect the current text mode and use it. This is the default.

2. The programmer can explicitly use 25x80, 25x40 or 43x80 (which gives 50x80 on VGA).

3. The programmer can give the user a choice on which mode to use.

Graphics mode MS-DOS display classes use compiler specific libraries. The library I have uses BGI (Borland Graphic Interface). The documentation indicates that there is similar support for Zortech's graphic library. An application can switch from graphics to text mode and back without losing the information in the user's windows. Besides the text mode features, the graphics display classes have support for graphic specific features like arcs, polygons, bitmaps, etc.

Editor's Note:

As this issue went to press, we were informed that Zinc Interface Library v3.0 is now available. According to a Zinc representative, version 3.0 addresses some of the problems noted in this User Report. New features include direct use of Windows bitmap functions; MDI support for Windows and DOS; new window objects such as toolbar, combo box, checkbox, radio button, and buttons with associated bitmaps; Zinc Designer adds a toolbar and access to all library features including user functions and validation routines. For more information contact Zinc Software Incorporated, 405 South 100 St. Suite 201, Pleasant Grove, UT 84062, (801) 785-8900, FAX (801) 785-8996.

David Brumbaugh is a project manager at Advanced Information Services, a systems integrator in Peoria, IL. He has been programming in C for over five years and in C++ for over a year. He can be reached by mail at 2807 N. Renwood Ave., Peoria, IL 61604.

While one MS-DOS application can support both text and graphics, you must recompile your program to support MS-Windows. You also must add a couple of `#ifdef` statements to call `WinMain` instead of `main`, and to redefine how colors are used. Other than that, all my MS-DOS applications, including graphic applications, ran without modification on MS-Windows.

The strong points of the class library far out-weigh the weak points. Because the class library is set up in a very logical hierarchy, it is easy to learn.

The field validation capability is one of my favorite features. All user input can be validated by the program. Most common validations are included and the programmer can define his own easily.

The library is fairly robust. Most of the errors I found in my programs were my own. When they weren't there was usually a fix on their BBS. The library is complete and generally well designed.

The only weak points that I found were in the MS-Windows mode. The first is that since it always displays its bitmaps one pixel at a time, bitmap displays in MS-Windows are very slow.

When I tried to work around this by using Windows bitmaps, I discovered that there is no obvious way to use Windows resources with Zinc. Zinc has its own version of resources, and the BBS contains some programs to convert Windows bitmaps to Zinc bitmaps and Windows icons to Zinc icons.

Finally, Zinc doesn't have any direct support for MS-Windows Multiple Document Interface (MDI). MDI applications

keep all the child windows of a main window confined within the boundary of the main window. Some programmer's will see this as a bigger problem than others.

The other major feature that comes with Zinc is the Zinc Designer. The Zinc Designer is a program that allows the programmer to draw windows, dialogues, menus, bitmaps etc. Using the Designer is faster than writing the equivalent source code. It allows the programmer to make better "Look and Feel" decisions.

I found several weak points with the Designer. Some of the features that Zinc supports, like radio buttons and check boxes, are not supported by the Designer. The menu items in the Designer are cumbersome to edit. My final complaint is that the Designer generates only binary objects, not source code. I would like to see both.

Documentation

The Zinc documentation consists of three books: The Programmer's Guide, The Programmer's Tutorial, and The Programmer's Reference. It also includes a Quick Reference Guide containing a list of the most common constructors, flags, event information, and a class hierarchy.

The Programmer's Guide provides a good overview of the concepts in the Zinc Interface Library. It is a short book that hits the most important points of the library. It also contains a user's guide to the Zinc Designer.

The Programmer's Tutorial is a clear and simple book to help the programmer get started. It is short enough to stay interesting. That is a major accomplishment when you consider that it not only contains lessons on using the Zinc Library, but a C++ and object-oriented design tutorial as well. The examples are excellent. They are clear and many are useful building blocks for your own applications.

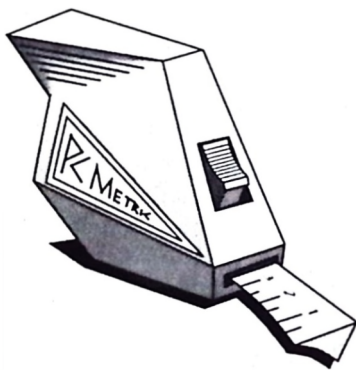
The Programmer's Reference is a well organized book that covers most of the classes that come with Zinc. It could be more complete in its handling of specific classes. For example, the width parameter in the `Line` method of `UI_DOS_BGI_DISPLAY` is ignored. I spent about two hours trying to find the bug in my code before I checked with technical support. They agreed that it should have been documented.

Support

I have found Zinc technical support to be absolutely terrific. I usually use the Zinc BBS for support. It is well organized and well maintained. It contains corrections, news, user contributions and additional examples. The message base provides contact with the people at Zinc and with other users. I usually search the message base and find the answer I'm looking for without having to post a question.

The Measure of a Great Program.

PC-METRIC™: The Measurement Tool For Serious Developers.



PC-METRIC is the software measurement tool that measures your code and identifies its most complex parts. So you can spend your time working in the areas most likely to cause problems.

PC-METRIC is fast, user-configurable, and includes a wide array of commonly accepted measurement standards.

Plus, versions of PC-METRIC are available to support virtually every popular programming language.

A Great Value By Any Measure.

PC-METRIC's price is only \$199, and it comes with a 30-day money-back guarantee. Multiple user discounts are available, as well as site licenses and complete source code.

Order Now! Call (503) 829-7123.



SET LABORATORIES, INC.
"Quality Tools For Software Craftsmen"
P.O. Box 868
Mulino, OR 97042
Phone: (503) 829-7123
FAX: (503) 829-7220

**COMPUTER
LANGUAGE
PRODUCTIVITY
AWARD
1990**

◆ Request 210 on Reader Service Card ◆

When I can't wait for the BBS I call Zinc's voice line. I've never had a long wait when I've called. The support people are very friendly, helpful and knowledgeable. They usually understood my problem better than I did and had helpful suggestions in addition to the answers to my questions.

Competition

Borland's ObjectWindows and Turbo Vision are the other C++ user interface libraries I am familiar with. Borland chose to use separate class hierarchies for MS-DOS text and MS-Windows user interface. That means that if you want to write one application for both MS-DOS and Windows, you have to write two separate programs.

ObjectWindows has more direct support for MS-Windows than Zinc. It includes classes that can use Windows resources, for example. But, it lacks the field specific editing features, like dates, that Zinc has.

Turbo Vision is a text mode user interface class library. It has features Zinc doesn't, like the *THistory* class that allows the user to keep a list of data entry choices. It also lacks certain features, like the extensive field support, that Zinc has. Turbo Vision also lacks the MS-DOS graphics support that Zinc has.

Generally, if you're doing strictly MS-Windows programming, Borland's OWL has a slight edge because it is strictly for Windows. Its bitmaps are displayed faster, it can use resources from Borland's Resource Workshop (or other sources) and there is full MDI support.

If you want to write strictly text mode programs, Zinc has a slight edge over Turbo Vision because it has more support for formatted and validated data entry fields. If you want to write MS-DOS graphics mode programs, Zinc wins hands down because neither Turbo Vision or OWL supports that mode.

If you want to write applications that can be used across all three platforms, I recommend Zinc because you'll be doing less rewriting of code. The Zinc library fulfills the promise of code reuseability much better than either ObjectWindows or Turbo Vision.

Conclusion

The Zinc Interface Library is a good value. There is no doubt that Zinc will save time for C++ programmers who write programs for the PC. It lets you program for three PC platforms in the time it usually takes to write code for one. The examples of C++ code will be of great help to the new C++ programmer. The library is an example of good Object Oriented Design.

I have only two suggestions for improvements. I would like to see a more complete reference manual. I also would like to see the Windows library have more support for the features of Windows 3.x.

I would recommend the Zinc Interface Library to any C++ programmer who wants to write applications for the PC. □

Notice

To Our Subscribers

Occasionally, *The C Users Journal* makes its mailing list available to vendors of products we think our readers will find interesting. Current subscribers receive free information in the mail from these vendors.

If you prefer that your name not be used in these mailings, please let us know. Just copy or clip this form and send it with your name and address to:

The **C** Users Journal™

The C Users Journal
1601 W. 23rd. St., Ste. 200
P.O. Box 3127
Lawrence, KS 66046-0127

WITH 'C' SOURCE

For **UNIX***
and **DOS**

*C-ISAM is a trademark of Informix Software..
Unix a trademark of AT&T.

'W'

MAKES CURSES obsolete!

- 'W' is a windowing library for 'C'
- 'W' gives you prioritized windows
- 'W' offers complete control of any terminal
- 'W' drives the DOS Memory map
- 'W' has line graphics and bells
- 'W' has soft keys
- 'W' offers 'hot key' control
- 'W' prints screens even in UNIX*
- Describing terminals is much easier with 'W' than termcap
- It's easy to port from curses to 'W'
- \$295 (U.S.), you get SOURCE for UNIX* & DOS

D-ISAM

C-ISAM* Compatible

UPCOMPATIBLE
WITH C-ISAM*

- Both use compatible function calls
- Both use compatible datafiles
- Both work the same files at run-time

WE ADDED:

- Extended types and natural ints
- Strings
- Improved file integrity checker

D-ISAM:

- A complete (B + tree) filing system
- Manages multiple keys
- Uses read only record locking

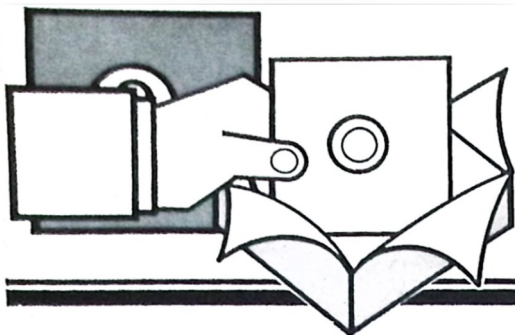
PRICE INCLUDING SOURCE:

- \$595 (U.S.) for UNIX* & DOS

BYTE
DESIGNS Ltd.

20568-32nd Ave. Langley, B.C.
V3A 4P5 CANADA
Toll-free 1-800-663-8547
PH: (604) 534-0722 FAX: (604) 534-2601

◆ Request 111 on Reader Service Card ◆



New Products

Industry-Related News
& Announcements



GUI Design Tool for OS/2 PM Adds 32-Bit C Code Generation

Gpf Systems, Inc. has released Gpf v1.3, an upgrade of its Gpf (GUI Programming Facility) tool for OS/2 Presentation Manager applications. The new version extends the platforms on which Gpf-generated applications can run by enabling developers to create native 32-bit applications. It also allows developers to define custom graphical objects for the user interfaces they design.

Gpf runs on OS/2 1.3 Standard Edition or later versions, and it supports Extended Edition for those applications using OS/2 DataBase Manager. The program requires a mouse and 1.5MB of available hard disk, and 6MB of RAM are recommended. Gpf is licensed on a workstation basis and is priced at \$995 for the first copy. For more information contact *Gpf Systems, Inc.*, P.O. Box 414, 30 Falls Rd., Moodus, CT 06469-0414, (800) 831-0017, FAX (203) 873-3302.



Quick and Cool PLD

Intel Corp has introduced a high-performance CMOS 22V10 compatible programmable logic device (PLD) running at 100-MHz count frequency with a 10 nanosecond (ns) propagation delay.

Intel's CMOS 85C22V10 PLD operates at 40 percent lower power than bipolar devices, thereby generating less heat. This translates into increased system reliability and decreased power supply requirements — important features for today's compact system designs.

Based on the largest-selling PLD architecture, the 85C22V10 is JEDEC compatible with standard 22V10's and is supported by current 22V10 software. Optional superset features included are programmable clock inversion and additional signal feedback configurations.

In quantities of 1,000 the PLCC and PDIP 10ns versions cost \$9.50 and the 15ns versions cost \$4.75. For further information call a local Intel sales office, or the Literature

Center at (800) 548-4725 (in the U.S. and Canada), or write for: *Intel Literature Packet #IP-87*, P.O. Box 7641, Mt. Prospect, IL 60056-7641.



Softaid Introduces 15 Mhz Z180 Emulator

Softaid has released a version of its UEM In-Circuit Emulator for Zilog's 15 Mhz Z180 processor. The Z180 offers a Z80 processor core with an extensive suite of on-board high integration peripherals. Recently Zilog introduced a 15Mhz version of the Z180, giving embedded designers access to more computing horsepower while maintaining software compatibility with their older Z80 code.

The UEM includes 131,072 hardware breakpoints, a 4KB deep real time trace buffer, performance analysis, and a memory access monitor. It comes complete with a Source Level Debugger that supports essentially all C and PL/M compilers.

Softaid also offers a free Guide to Developing Z180 Applications, for those

designers needing a little more information than provided in Zilog's sometimes cryptic data sheets.

The 15 Mhz Z180 UEM costs \$6500.00 and is available from stock. For more information, contact *Softaid, Inc.*, 8300 Guilford Rd., Columbia, MD 21046. (800) 433-8812.



Demo Maker/Player Programs for Software Systems Running Under X

Non Standard Logics, Inc. (NSL), has released a set of programs that allow developers working in X to create demos of their software quickly and inexpensively, with unlimited distribution of the demos at no cost beyond the initial purchase price.

Designated XDemoMaker and XDemoPlayer respectively, the two-program set employs an OSF/Motif graphical user interface, but demos produced are independent of the GUI and run under any X terminal graphical user interface used with the application. The program incorporates editing tools for fine-tuning the demos and for including text to amplify on the activities displayed on the screen. Complete or partial sequences can be repositioned within a demo by means of "Cut-Copy-Paste-Duplicate" commands.

XDemoPlayer is licensed to purchasers of XDemoMaker for use on specific platforms with unlimited copying rights, enabling developers to distribute their demos with the playing program in quantity. A standard 1.44MB diskette accommodates both the XDemoPlayer software and a demo of several minutes' length. Longer demos can be circulated on tapes and compact disks.

Single unit price for XDemoMaker is \$6,000. XDemoPlayer is licensed for \$4,000 for the first platform with discounts available for additional platform versions. Unlimited copying rights are included with the player license. The programs are now available for

IBM, DEC, Sun and Hewlett-Packard workstations. For more information, contact *Non Standard Logics, Inc.*, 4141 State Street, Suite B-11, Santa Barbara CA 93110, (805) 964-9599, FAX (805) 964-4367. In Europe, contact *Non Standard Logics SA*, 57-59 Rue Lhomond 75005 Paris, France, (33) 1 43 36 77 50, FAX (33) 1 43 36 59 78.



New Release of Speededit

Bradford Business Systems recently released SpeedEdit vA.04, a programmer's text editor for MS-Windows, X-Windows, DOS, OS/2, MPE-XL and other operating systems and platforms. This version adds support for SunOS and Hewlett-Packard's MPE-XL. Under each platform, save MPE & MEP-XL, SpeedEdit comes in one of two forms, a character version to operate on standard terminals or non-windowed consoles and a Windowed version. The SpeedEdit system sells for \$295 on all IBM-PC based platforms and \$395 on single user UNIX workstations. Multi user UNIX systems are priced according to the system. For more information contact *Bradford Business Systems, Inc.* at 23151 Verdugo Drive, Suite 114, Laguna Hills, CA 92653 (714) 859-4428, FAX (714) 859-4508.



Multilingual Software Toolkit

Frontier Software Services has released the Translator's Apprentice, a programmer's toolkit which allows the programmer to write programs for multiple languages (French, Spanish, Russian, etc). Using the toolkit, programmers can create multi-lingual programs which start up in a user's choice of language and switch from one language to another at the press of a key.

The package simplifies the tasks of the analyst, programmer, and translator by providing a mechanism for the translation and update of textual material in an application without requiring changes to the program for each translation.

The toolkit costs \$175 for the Text Manager and Access Functions, including source code and a 30 day money-back guarantee. Site licensing is available. For further information contact *Frontier Software Services* at 31 Mystic Avenue, Winchester, MA 01890 (617) 729-2161.



Debugging and Data Capture Tool

Paladin Software, Inc. is now shipping an upgraded version of DataScope™, the communications debugging and data



Visual Program Design and Code Generation Tool for MS-Windows

ProtoView Development has released ProtGen 2.1, a visual program design and code generation tool for Windows application development. Similar to ProtoGen 2.0, sold by Borland International, version 2.1 features a menu designer and dialog linking interface. It also generates code for both ANSI C and Borland C++ Object Windows.

Using a C++ class object, ProtoGen 2.1 also allows C++ programmers to tap into the Proto View dynamic link libraries to access advanced data entry and validation capabilities. Masked input, date, currency,

numerics and table collection controls and more, give a sophisticated look and feel with no coding effort.

ProtoGen's price is \$199.00, but is now available at the introductory price of just \$49.95. Upgrades for current users are also \$49.95. It requires a PC with 80286 or higher processor running Windows in protected mode. Code generated by ProtoGen is royalty free. For further information, contact *ProtoView Development Corporation*, 353 Georges Road, Dayton, New Jersey 08810, (908) 329-8588.

capture tool with applications in the computer programming, manufacturing, industrial automation, and multimedia industries. Version 2.0 saves time and money, eliminates guesswork by allowing the user to apply powerful display and search tools to ordinarily invisible transmissions, and provides an alternative to expensive hardware line monitors.

DataScope is the only serial line monitor that includes context-sensitive Hypertext, Hypersetup, and user-alterable multitasking window displays. Version 2.0 offers a user-friendly, "windows-like" pull-down menu interface.

For further information contact *Paladin Software, Inc.*, 3945 Kenosha Avenue, San Diego, CA 92117, (619) 490-0368, FAX (619) 490-0177.



Professional Input Validation for MS-Windows

MantaSoft Partners have released InControl, a new set of DLLs, with source code available, that will enable programmers to include professional input validation in their Microsoft Windows 3.0 applications. These DLLs are compatible with Microsoft Windows SDK, Borland C++ 2.0, Borland Turbo Pascal for Windows, Whitewater's Actor and Microsoft Visual BASIC.

InControl Toolbox defines thirteen new classes of controls for Microsoft Windows. The controls are broken into two logical groups: display and input. Display controls are used to display information to a user of a Windows application. Input controls are used to control the information that a user enters into a Windows application.

InControl Toolbox contains two display controls. These are the Time control and the Date control. It contains 11 input

controls. The input controls are broken into three groups: Formatted, Numeric and Free Format. The Formatted group contains controls which allow the input of Zip Codes, phone numbers, Social Security numbers and dates. There is also a programmer defined Formatted control that allows the programmer to define, at a character-by-character level, the allowable input for a given control. This control uses a dBase-like command string to describe the allowable input, if any, at a given position in the input control.

The Numeric group contains controls that allow the entry of integers, floating point numbers and dollar amounts. Each of these controls allows the entry of negative numbers and has an option which allows the programmer to set a legal range of values.

The Free Format group consists of two general entry controls. They are called general entry because they accept any input and, upon losing focus, try to determine if that input corresponds to a legal value. The first of these controls is the Free Format Date control. This control will try to determine what date the user meant from the given input. For example: a value of "F392" would yield the result "February 3, 1992" or "02/03/1992" depending on a style flag. Other allowable inputs are "Next Week", "Last Friday", "Today" and "Yesterday".

The second general entry control is the Regular Expression control. This control uses a Grep-like regular expression to determine if the value entered by the user was a legal value.

For each of the input controls, a programmer can assign a function that is called when the control loses focus. This routine can be used to further validate the data entered or to immediately store the entered value into a variable in memory. This function notifies the control, through a



C// for C++

Subtlesoft announced C// for C++: the C extension injecting real-time parallelism into a single C program, now combining the power of C// and the flexibility of C++. An unlimited number of run-time created, separable processes smoothly cooperate on common resources, self-parallel functions, queues, lists, events, timeouts. All the C++ communications and classical synchronization mechanisms are available, along with a powerful set of new C// weapons, including the new C// class of semiautomatic variables, run-time control variables, double access to process arguments, stack monitoring, private stacks, processes with no stack, offsets, etc. Dynamic priorities and scheduling facilitate interprocess cooperation. Precise handling of external events

through the C// driver, user-programmed ISRs, urgent process executions, and the complete resolving of the problem with DOS and BIOS non-reenterability make C// for C++ a useful real-time tool.

C// for C++ provides good memory management and performance. It provides a natural platform to pure object-oriented programming in addition to the C++ mechanisms. Real-time projects, communication and simulation software, internal multitasking environments, advanced control software are effectively implemented in C// for C++.

For further information contact *Subtlesoft International, 4344 Bristol Street, Pittsburgh, PA 15207; (412) 682-3934.*

objects. Object Professional for C++ (OPC) is a straight port of the user interface objects from Object Professional for Turbo Pascal. OPC includes high-level objects such as text editors, dialog boxes, scrolling data entry screens, help systems, and more. The user interfaces are ready-to-use, and built-in calls allow programmers to customize the behavior. OPC does not use event-driven programming. Applications built using OPC will run nicely in 640KB 8088 based PC's.

OPC includes utilities to help build menu systems and data entry screens. The utilities support interactive design and testing and then automatically generate source code. OPC includes full source code, 1,300 pages of documentations, pop-up help, and plenty of example and demo programs. No payment of royalties is required. OPC requires Borland C++ 2.0 or 3.0 Object Professional for C++ costs \$249. For further information contact *TurboPower Software, P.O. Box 49009, Colorado Springs, CO 80949-9009, (800) 333-4160.*

return value, whether it considers the value to be valid or invalid.

Each control can notify the user of error conditions in two forms. A style can be set that causes an audible beep whenever an error is detected. Alternatively, a visual text message can be displayed immediately below the current input control which more fully describes the error. For example, if a user enters a '9' in a Formatted control in a location where the maximum value has been set to '5', the control would display a message informing the user of the range violation.

InControl Toolbox is available directly from MantaSoft Partners and retails for \$179, or \$249 with complete source code. Orders will also be taken by GUI Clearing House, which can be reached at 1-800-522-4624. For more information please contact *MantaSoft Partners, P.O. Box 203551, Austin, TX 78720, (512) 335-3497, CompuServe 70314, 1445.*



Programming Library With Added Memory Capability

Library Technologies announces the addition of EMS/XMS/virtual memory capabilities to their programming library C-Heap for the Microsoft C and Borland C/C++ compilers. The 51 new functions give the programmer low-, medium-, and high-level functions for utilizing EEMS 3.2, LIM 4.0, XMS 2.0, and virtual (disk space) memory, with no 64KB limit on memory block size. The most powerful, high-level functions provide an automatic swapping mechanism (with locking capability) to completely remove the burden of managing

EMS/XMS/virtual memory from the programmer, making the use of EMS/XMS/vmem an extremely simple matter. At all function levels, the programmer specifies whether EMS or XMS is to be used preferentially, but the functions use whatever is available. If neither is available, or when EMS/XMS is used up, the medium- and high-level functions can use disk space automatically. The user of disk space can be selectively disabled, though, or memory can be allocated specifically from disk space if desired; thus, the programmer can be sure that memory is being utilized exactly as he wishes. All virtual memory I/O occurs through from zero to four disk caches of up to 16KB each in size, which the programmer may specify be allocated to dramatically increase the speed of virtual memory I/O. With the high-level functions, a "dirty bit" can be cleared to avoid writes to EMS/XMS/virtual memory when the data has not been changed, which can greatly speed execution. All functions are compatible with malloc(), and all memory models but the tiny model are supported.

These functions join the other 500+ assembly-language functions already in C-Heap, which deal with DOS memory management. The cost of C-Heap is \$199, or \$399 with source code. For further information contact *Library Technologies, P.O. Box 56031, Madison, WI 53705-9331, (800) 767-4214.*



Object Professional for C++

TurboPower Software announces Object Professional for C++, a powerful, time-proven library of text-mode user interface



Shortcut for Rapid Application Prototyping

Cadre Technologies Inc. announces the availability of ShortCut™, a rapid application prototyping tool based on Cadre's TeamworkR family of CASE products. Developed by Cadre's exclusive Austrian distributor, Computer & Software Engineering (C.S.E.), ShortCut incorporates a graphical user interface (GUI) editor, a database access library, a compiler, and a rule-based expert engine. Working together these components enable application developers to capture, analyze and validate end-user requirements before designing and implementing the actual production system. In addition they allow developers to generate a complete and functional application prototype and enable developers and potential end-users to interact with this prototype well in advance of the development of the code.

With ShortCut, developers can understand end-user requirements from three important application views: the user interface; functionality and control; and database access, which works with SQL databases such as Ingres, Oracle, and Sybase.

Priced at \$7,995 for a base configuration that includes a GUI editor, a compiler, and an expert runtime system, ShortCut is available immediately on Sun, DEC, IBM, and HP workstations. For further information contact *Cadre Technologies, Inc., 222 Richmond Street, Providence, RI 02903, (401) 351-CASE, FAX (401) 351-7380.*

Database Independent Development Environment

Convergent Solutions, Inc. (CSI), has released its database independent development environment, CS/ADS Release 6.3, for use with ShareBase Database Servers attached to Sun 4 and Pyramid MIsServer computers.

CS/ADS is a tool set for professional programmers. It provides a comprehensive high-productivity development kit for building complex business applications and information systems. A significant benefit of CS/ADS is that it allows applications to be built that are database independent, yet still incorporate the full power of SQL for the specific RDBMS.

CS/ADS applications that are built for ShareBase can be easily moved to other RDBMS's by using CS/ADS for ORACLE or CS/ADS for Informix.

The centerpiece of CS/ADS is a fourth generation programming language (4GL) that combines high-level constructs and powerful abstract datatypes with the programming structure of traditional third generation languages. The development environment includes a dynamic data dictionary that extends the data definition beyond the database schema by including editing, verification, and formatting rules for data fields. Also, CS/ADS comes with a suite of interactive, developer friendly utilities such as an advanced WYSIWYG screen painter and 4GL code generator, all of which are integrated to both the data dictionary and the RDBMS schema.

CS/ADS for ShareBase III will be initially available for Sun 4 and Pyramid architectures. The software will be ported to other CS/ADS supported platforms later this year.

For more information contact *Convergent Solutions, Inc., 100 Metro Park South, Lawrence Harbor, NJ 08878, (908) 290-0090, FAX (908) 290-1494.*

Program Language Translation

Shannon Associates' METAMORPHOSIS is a generic utility program which facilitates the transformation of any syntactically reducible character-oriented file to other forms. Given the syntactical definition of the source and target languages, METAMORPHOSIS will translate source programs from one language to another; ie: FORTRAN to Ada, JOVIAL or Ada, any language to C, dialect conversions, etc.. METAMORPHOSIS also functions as a cus-

OS9 for Motorola Single Board Computer

Microware Systems Corporation now offers an optimized version of OS-9 for the Motorola MVME167 single board computer.

This latest version of OS-9 for Motorola microprocessor-based products allows designers to take full advantage of the 32-bit M68040 processor, while providing support for the on-board serial, SCSI and Ethernet hardware.

The OS-9/167 Development Pak includes a number of new OS-9 device drivers for the next generation I/O peripherals included on the MVME167 board family. These include new SCSI drivers for the NCR 53C710 controller which support Common Command Set flexible and hard disk drives and tape units. Additional drivers are included to support the on-board real-time clock and new CD-2401 serial I/O controller.

Full Ethernet support is provided by means of the OS/9 Internet Support Package (ISP) and new device drivers for the Intel 82597 Ethernet Controller. ISP allows an OS-9/167 Development Pak system to remotely login and transfer files between an MVME167 and UNIX or DOS nodes on an Ethernet network. Support for BSD socket-

based interprocess communication is also provided.

The OS-9/167 Development Pak also includes a full set of resident development tools designed to jump-start application development. These tools include a full K & R C Compiler, Macro Assembler and Linker, User State Debugger, (mu)MACS full screen editor, Shell Command Interpreter, and numerous utility programs.

OS-9/MVME167 is available in two versions. The OS-9/167 Development Pak includes the OS-9/167 Real-Time Operating System modules and device drivers as well as the full suite of development tools. Cost for the Development Pak is \$3,000. The OS-9/167 Run-Time Pak provides only the OS-9 Real-Time Operating System modules and is intended to provide target system functionality. Quantity one pricing for the Run-Time Pak is \$1,500. Contact Microware for multiple-copy licensing information. Both packages are available now.

For more information contact *Microware Systems Corporation, 1900 NW 114th Street, Des Moines, IA 50325-7077, (515) 224-1929, FAX (515) 224-1352.*

tom compiler, assembler, macro processor, graphics language processor and report generator. Further, METAMORPHOSIS facilitates reformatting of data base files and analysis of natural language, grammar, sequential and parallel procedures and computational signatures.

In addition to generic METAMORPHOSIS, preconfigured METAMORPHOSIS translators including FORTRAN IV to C PL/I (Subset G) to C, and CMS-2M to Ada are available for immediate delivery.

Generic METAMORPHOSIS, FORTRAN IV to C, PL/I (Subset G) to C and CMS-2M to Ada sell for \$387, \$87, \$87, and \$134 each, respectively. All execute on the IBM PC, PC/XT, PC/AT and compatibles with minimum 416K RAM, monochrome monitor, two 360K bytes 5.25" DSDD floppy drives, optional printer and MS-DOS / PC-DOS 2.0 or revisions.

Contact *J.H. Shannon Associates, INC., P.O. Box 597, Chapel Hill, NC 27514 (919) 929-6863.*

New Name for Raima Product

Raima Corporation announces that they are changing the nomenclature of db_VISTA, the high performance database

management system, to Raima Data Manager. This move is in line with the company's new corporate image campaign of relating its product names to their specific technology.

The Raima Data Manager DBMS is best known for its high performance in applications development, and for its combined database technology of relational and network models.

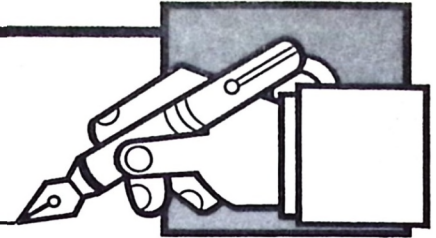
Raima began the new naming campaign last June when they introduced Raima Object Manager, a storage class library targeted in the C++ and object-orientated marketplace.

The name change to Raima Data Manager accompanies a new update release, version 3.21. The update does not present new features, but includes fixes to the current 3.2 version.

Raima Data Manager will continue to be marketed as both a stand-alone product, and as a full system accompanied by db_QUERY and db_REVERSE. db_QUERY is an SQL-based query and reporting module; db_REVERSE is a sophisticated database restructuring tool.

For further information contact *Raima Corporation, 3245 146th Place S.E., Bellevue, WA 98007, (800) 327-2462, FAX (206) 747-1991.*

We Have Mail



We ask that letters with code listings be submitted in an ASCII text file on an MS-DOS formatted disk. Providing us an electronic copy of the code will prevent typographical errors that might result from optical scanning or re-keyboarding.

Dear Sir,

I see that Randall Bart has raised the question of C++ vs. ++C. My own understanding is that the language is so named because every time you think you've finished learning it Bjarne adds a new feature.

Yours faithfully,

Scott Wheeler
BMT Research, Orlando House
1 Waldegrave Rd.,
Teddington, Middx
UK TW11 8L2

Wish I'd said that first. — pjp

Gentlemen;

It seemed good to take this opportunity to thank you for the work your staff has done this past year, and the appreciation many of us have (who don't write) for the quality of *The C Users Journal*. As a CPA, and accounting software developer, each new monthly issue frequently discusses and answers a current concern of mine. I can't tell you the number of times an article appeared at exactly the moment I needed the information. True, many articles are over my head and I marvel at how so much can be written about something I know so little about.

It also seemed good to nominate Mr. Leor Zolman as CUJ Writer of the Year. This in no way slights the efforts of the others, but my vote is based on the following criteria:

Appropriateness and general usefulness of subject: A.

Scope and technical considerations covered: A.

Ability to adapt subject material to other applications [portability]: A.

Tutorial content of material and usefulness in teaching C language concepts and capabilities: A.

Humor and 'tongue in cheek' writing style (a spoonful of honey makes the medicine go down): A.

Future subjects that would be of interest to me and perhaps others are suggested for your consideration, and include:

DOS TSRs: For example, how would one adapt Mr. Robert Bybee's article "A Portable VMS-Style Input Line Routine" to function as a TSR, similar to Peter Norton's NDE referenced in the article.

Keyboard I/O: A routine for polling the keyboard buffer. CUJ had an article on this subject some time ago, but it either did not go far enough or was beyond my ability to understand. The scope of the routine would be in two parts: (1) the ability to capture user keystrokes from an application and log them into an ASCII script file (perhaps subsequently edited by the user), and (2) the ability to call this routine and load the named script file into the keyboard buffer. In this way, users should be able to run repetitive applications or set up a series of applications (reports) to be chained together and run as a batch.

Batch File Compilers: Sometime ago I downloaded a batch file compiler from CompuServe, written by Douglas Boling (PC Magazine Ziff Communications). The language concepts I think that are required should make an interesting article. (I would love to know how this program works.) The input file consists of a standard W S batch file which is then compiled on the fly and a .COM file written to disk. How is this done?

Help Compilers: Beginning with standard documentation files (ASCII) as might be produced by a word processor, how would one build a compiler to translate ASCII files into an indexed binary format. The help utility program would access a file name passed on the command line, display an index for user selection, then handle paging and text display with keyboard control. Based on Leor's current article, this should be a piece of cake for him to take on as his next project. (You're welcome, Leor.)

Enclosed with this letter is my subscription renewal and disk order for processing.

Thanks again for your quality work in past years and may this year generate

even more success for R&D Publications.

William C. Moench, CPA
North Gate Software
29204 Knickerbocker Rd.
Bay Village, Ohio 44140

It's always nice to hear when we do something right. Thanks for writing. — pjp

Leor responds:

Gee, if they'd given me report cards like that when I was a student at MIT, perhaps I wouldn't have dropped out! On the other hand, this kind of feedback probably means more to me than any real report card I've ever received. Thank you much.

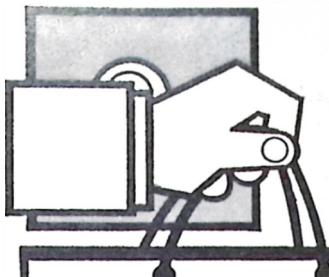
I find it difficult these days to sit down and write a program just as an intellectual exercise; usually, I have to have an actual need for the program before I'm motivated to write it (that's partially why I have resisted getting into programming Windows; the other reason, a lack of elegant development tools, however, seems to finally be lifting...) Most of the material for Illustrated C derives from applications I've written and put into service pretty much as they appear in the column and, now, the book as well.

Incidentally, when bugs pop up, it's usually because the program hadn't gotten enough use yet. I'm too scatterbrained to debug by inspection; I have to debug by "using" — yes, otherwise known as trial-and-error. Plus, I'm a perfectionist. If nothing else, that makes for an entertaining development cycle. Well, in the immortal words of our distinguished publisher, "OK, I'm done."

Dear Mr. Phillips:

I very much enjoyed Part 5 of your series on Image Processing. It has stimulated some creative thought processes in me and for that I thank you. I did notice one problem with your examples which caused me to write this letter. The discussion below will clarify the situation.

The Kirsch, Prewitt, and Sobel transform sets are designed to detect edges in only one of the eight compass directions. As such, they can only be applied to the original image and can not be



Programmer's Market

C™ MPW (\$295); DOS Vol 1, 2 & 3 (Any volume \$125; any 2 volumes \$225; All \$300); Sun UNIX (\$495)

Programmer's Toolbox

30 Day Money Back Guarantee

Dialects: Microsoft C6.x; MPW C; Think C; Turbo C; Unix C; ...

Unlimited number & sizes

CLint™ - Syntactically check one or more files & generate ANSI function prototypes; CFlow™ - Determine program organization, function and file interdependencies and runtime library contents; CPrint™ - Reformat/beautify C & C++ source code; CXref™ - Cross reference and check symbol usage, and identify unused files; CDecl - Translate and compose C declaration statements; CHillite™ - Highlight & print C source files (MPW only); pArc™ - File archiver with great compression (DOS only); pLn™ - UNIX symbolic links in DOS; PMon™ - Program profiler (DOS only); Cpp-flexible ANSI C preprocessor; plus many more...

SPECIAL

Buy 2 PC volumes and get 3rd free!

Call: (510) 770-0858
MasterCard Visa Accepted

MMC AD Systems
Box 360845
Milpitas, CA 95036

◆ Request 152 on Reader Service Card ◆



Writebar Barcode Products

Wilsoft carries a complete line of bar code software and hardware for your every need. All major bar code types supported. Call the most experienced company in the bar code field today. DOS and Xenix/Unix support. Portable readers too! VISA/MC/AMEX

109 NW 22 Street
Ft. Lauderdale, FL 33311-3834
(305) 779-2720
(305) 763-3096 Fax
(800) 765-4114 Sales

◆ Request 115 on Reader Service Card ◆

PostScript From C!

c_pslib

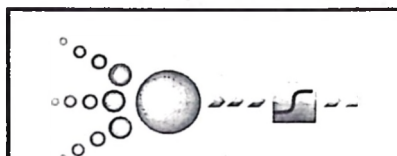
- ✓ Complete Set of Graphics Functions
- ✓ Text Functions for fonts, lines, paragraphs
- ✓ Downloadable Font support
- ✓ Encapsulated PostScript File support

c_psFontInfo

- ✓ Stringwidth Calculations
- ✓ Font Reencoding
- ✓ Kerning, Ligatures, & More!

Site License \$595 each
Single User License \$195 each
ANSI and K&R Source Code Included
Barton Creek Software
2222 Western Trails Suite 106, Austin, Texas 78745
512-441-8354

◆ Request 471 on Reader Service Card ◆



Neural CASE™

More than a shell - a complete neural network application development environment. Train, test, and deliver your application using one product. Supports four network paradigms: BPN, CPN, FLN, and SOM. Object-oriented approach - easily create and configure I/O and network objects using a powerful menu-driven GUI. Weights generated can also be used by the Neurocomputing Library. Introductory price: \$199.

Neurocomputing Library

A library of C neural network functions featuring: twelve popular neural network paradigms, multiple networks and paradigms in a program, example programs, and royalty-free distribution of applications. Supports Turbo C 2.0, Turbo C++ V1.0, Microsoft C 5.1, or Zortech C++ (please specify when ordering). \$179. Library with source \$379.

NeuroSym® Corporation
P.O. Box 950683 • Houston, Texas 77098-0683 • (713) 523-5777

◆ Request 327 on Reader Service Card ◆

PAUWELS CATALOG

A catalog of DOS & OS/2 software on your computer. Cross - referenced on product name, category and editor. Monthly updates.

The PAUWELS CATALOG is distributed as shareware (available on CompuServe - !BMAPP - file PAUWEL.EXE)

A tool for programmers, MIS departments and software dealers.

Pauwels & Partners Software

◆ Request 209 on Reader Service Card ◆

Optimize Your Object Code Segment Consolidator

An inter-object module optimizer

Convert large code model code to mixed model, delete dead code, convert to register parameter passing, convert to pascal stack clearing convention. No source code modifications required. Output is in ready to link object module format. Supports C language MSC 6.0, Borland C++ 3.0, Assembly generated and runtime library object modules. 30 day money back guarantee. Only 145.00.

Optimite Systems
1000 Singleton Blvd
Dallas, Texas 75212
(214) 745 1301

◆ Request 188 on Reader Service Card ◆

If You Program In C You Need This Make Utility

Programmer's SUPER-MAINT™

Professional make utility automatically writes your make and response files for you, remembers your command flags, the make file name, and more! Leaves all but about 2K of your memory for compilers to use. 30 day money back guarantee.

New Version 3 has Smart Library Maintenance, on-the-fly response file building, multiple models built with one command. "Point and shoot" at code files you want in your program, multiple setups, user configurable. Control how your program is built from the command line using simple mnemonic flags. Still **only \$55** (plus 3.00 edb, NY residents add sales tax).

EmmaSoft
PO Box 238
Lansing, NY 14882
Voice: (607) 533-4685
BBS: (607) 533-7072

◆ Request 197 on Reader Service Card ◆



FIGHT PIRACY & PROTECT YOUR PROGRAM \$\$\$'s!

Since 1986, companies worldwide have been choosing Az-Tech security products. If you demand the strongest protection available, why not choose one of these "proven leaders":

- EVERLOCK Copy Protection
- EVERTRAK Software Security
- EVERKEY II "The Lock"

For IBM and Compatibles.
30 day money back guarantee.
Free info and demo disk available.

AZ Az-Tech Software, Inc.
201 East Franklin, Ste. 11
Richmond, MO 64055
(800) 227-0644 FAX (816) 776-8398 (816) 776-2700

◆ Request 245 on Reader Service Card ◆

cascaded. The reason for this is that each individual transform is designed to REJECT and REMOVE information from its output which does not correspond to a gradient in the direction of interest. I noticed in the photo examples that the images had a distinct bias in favor of one gradient over the other. For example, the right edges of dark objects are shown but the left edges of dark objects are missing. This is due to the fact that you have applied the transforms in cascade. The correct use of these transforms dictates that they each be applied to the input image and then the results analyzed.

It should be noted that it is not possible to merely sum the outputs from the eight transforms of the Kirsch and Sobel sets since this would result in values of zero. With these sets however, it is possible to accumulate the absolute value of the transform output. Since the transforms with opposite compass directions create symmetrical output, it is only necessary to use the absolute value of four patterns. The strong negative output of the one direction matrix provides the detection capability previously provided by the opposite transform. The output of the eight Prewitt transforms may be summed to non-zero values but it may be advantageous to use the same feature of symmetry to avoid redundant calculation.

I hope that this comment is not too brief and that you find the information useful. Again thanks, and I look forward to your next article.

Sincerely,

Frank Evans
Intellex Vision Products

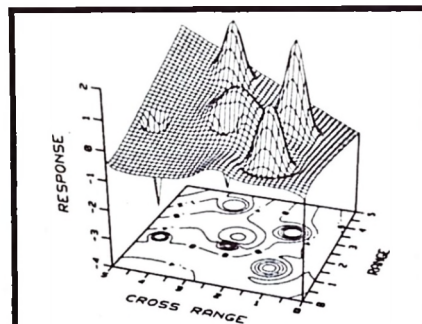
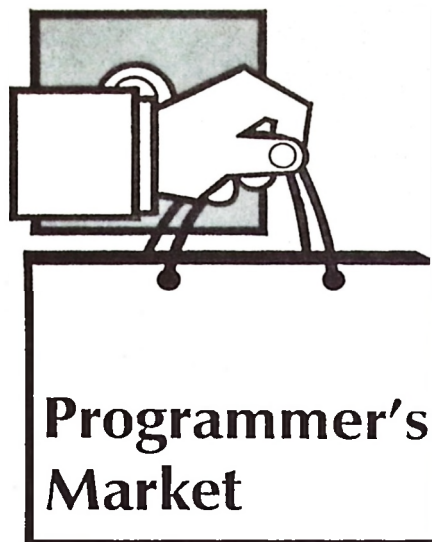
Dwayne Phillips replies: Thanks for the comments.

Gentlemen;

I have just gone through the process of learning how to use TLIB, including 15 minutes on "hold" in a telephone call to Borland. As an ex-professor (of a technical subject), and as a novice C user, I feel it is necessary to make two comments and to editorialize on Mr. Pugh's answer to Mr. Sam LeFevre, of Idaho Falls, as presented in the October 1991 *C Users Journal*.

Mr. LeFevre introduced himself as a "novice Turbo C user," and asked for guidance in setting up his own library of functions. Mr. Pugh correctly advised him to use the TLIB program supplied with Turbo C. But then,

1) went on to hide the really important advice (global variables) in typical and unnecessarily complicated C (for a "novice") programming, and



New! DIG for WINDOWS! Save Time!

Contour, Bar, Pie, X-Y, Log, Spherical & 3D Fishnet graphs are autoscaling. Dynamic 3D motion of models. Fast polygon fill. Write one routine to support CGA, EGA, VGA, Herc, PostScript & Dot Matrix. Laser, Plotters at full resolution. NO ROYALTY! For C, C++, QuickBASIC, BASIC, FORTRAN and Windows. Prices begin at \$180!

CHIRP TECHNICAL SERVICES

Call For Details! (619)632-9510

SCALABLE FONTS

EXCLUSIVE! Integrate the *FastFont Typeface Manager* object library into your application to add fast, WYSIWYG, on-the-fly printer and display fonts. Includes the three most popular hinted font families. 300+

fonts available. Only \$495 with NO ROYALTIES.

Optional Printer Driver Platform, Font Conversion Utilities, et al.



Ancier

Technologies, Inc.
5964 La Place Ct., #125
Carlsbad, CA 92008
Ph: 619/438-5004x132
Fax: 619/438/6898

◆ Request 184 on Reader Service Card ◆

Put GRAPHICS in your UNIX programming

VGA, EGA & Hercules
UNIX PLOT(3) subroutines for
SCO XENIX & UNIX, 386/ix,
AT&T System V/386, VENIX

- Circles, rectangles, fills, user-defined shapes and more
 - Coded with integer arithmetic
 - One binary supports all 3 modes
 - Most graphics cards supported
- \$80 + \$3 shipping/handling: single user
\$400 + \$3 shipping/handling: developer

Certified Scientific Software

P.O. Box 802168
Chicago, IL 60680
312-247-2442

◆ Request 208 on Reader Service Card ◆

TCP/IP

NETWORKING PROTOCOLS

Add them to your System
Designs with:

FUSION Developer's Kit

- FUSION TCP/IP protocol suite
- Flexible architecture - C source code
- Used in hundreds of process control, embedded systems, and end-user designs
- Full support with consulting and porting services

Call: 800-541-9508

805-485-2700

Fax: 805-485-8204



Network Research

◆ Request 494 on Reader Service Card ◆

QualBase™

For DOS

C++ Class Library For Class Designers

Increase software development productivity by choosing from a wide variety of efficient, reusable, portable, extensible, and maintainable classes. Source code, test suites and documentation included.

\$175.00

Qualware™

(714) 259-1322

◆ Request 382 on Reader Service Card ◆

Printer Graphics Libraries

PGL ToolKit is a set of easy to use libraries for generating device independent high resolution printer graphics on most popular printers. Provides 80+ functions to create vector drawings and/or print bitmapped images. Provides full control of printing (margins, layout, size, resolution, etc.) and supports parallel and serial port interfaces. Includes full support for C/C++, FORTRAN, Pascal, Basic, Clipper, & Assembly plus 32-bit support for C and FORTRAN.

NO ROYALTIES!
Only \$145 (includes all language support)

AnSoft, Inc.
8254 Stone Trail Court
Laurel, Maryland 20723 USA
Voice/FAX: (301) 470-2335

◆ Request 412 on Reader Service Card ◆

Motif Developers

User interface toolkit for Motif programmers. Set of widgets provide data byte specific field entry with value checking. Save 90% development time and difficulty. Circle reader service card for informative flyer.

MARLAN SOFTWARE

◆ Request 145 on Reader Service Card ◆

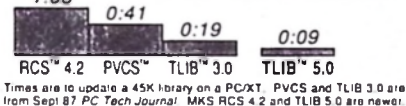
PROSYS — the Project Development Tool

DOS programmers!
Slash interface development time for your next text-mode "C" application. Complete interface development including screens, menus, overlapping windows, context sensitive HELP, report writing, and even program installation.
Design screens and menus with our easy to use screen painter. Add data I/O fields to screens with automatic data validation by internal or user supplied functions. Project reports simplify internal documentation. String externalization supported (foreign language support). Rapid project prototyping, with automatic structured code generation directly from the prototype. Built in seamless security system.
Linkable libraries include functions to easily manage the user interface. One simple function call is all it takes to do screen input, validate the fields, and request HELP.
Price \$99.95 complete.
Microsoft® "C" (ver 5.0 or later) required. Specify disk format. \$3.00 S&H. PA add 6%. No royalties. 30 Day money back if not satisfied. Check, MO to:

Killdeer Software, Suite 192
4676 Broadway, Allentown, PA 18104
215-434-4036

◆ Request 117 on Reader Service Card ◆

TLIB™ is FASTEST!



TLIB™ is BEST!

"Do not be fooled by the fact that this is the least expensive of the five packages reviewed here - TLIB has features and power to spare"
John Rex, Computer Language
"TLIB is a great system" J. Vallino, PC Tech J

• Full-Featured Version Control for Software Professionals. Check-in/out locking. Branching. Keywords. Wildcard and list-of-file support. Can merge parallel changes and undo intermediate revisions. Network and WORM support. Mainframe compatible deltas for Pansophic, ADR, IBM, etc.. Integrates with Opus™ MAKE & Slick™ MAKE.

MS-DOS \$139, OS/2 \$195 + shipping Visa/MC 5 station LAN license \$419 (OS/2 \$595), call for other sizes

BURTON SYSTEMS SOFTWARE
PO Box 4156, Cary, NC 27519 (919) 233-8128

◆ Request 430 on Reader Service Card ◆

C Object Programming

The COP manual prescribes a coding style for incorporating objects into your ANSI C applications. COP objects provide for encapsulation, single and multiple inheritance, polymorphism (virtual functions), and are extensible. Now you can port C++ OOP designed applications to C. Includes DOS formatted 3.5" and 5.25" diskettes with header and source object templates, only \$25 (add \$10 foreign delivery).

The Loose Data Binder, LDB v1.6, is a ANSI C persistent container object library written with COP. Includes 100+ page manual and diskettes with source (requires COP above), only \$35, add \$15 for foreign delivery.

PSW, P.O. Box 10072
McLean, VA 22102-8072 USA
(703) 759-3838

◆ Request 133 on Reader Service Card ◆

"An unusual name for an unusual product. MITUI is a set of classes that conform to the CUA standard taking the hard work out of Interface Development."

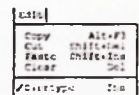
Program Now (UK), October/91

If your C-based software development should be...

- ... Windows and OS/2's PM look-alike and conforming to IBM's CUA/SAA standard;
- ... based on a small and high-level API made up by less than 70 functions (instead of over 500 provided by the competition);
- ... able to run on every PC/MS-DOS-based computer (not just on fast 386s & 486s);
- ... based on highest quality, portable tools, then you need

MITUI

The first high-level CUA API®



Available for Borland and Microsoft C & C++ compilers from:

MBI - Mayer & Bunge Informática Ltda.

Address: Caixa Postal 22.201 / 01498 São Paulo (SP) / BRAZIL
Phone/Fax: (+55) (11) 872-0497 Phone: (+55) (11) 872-4758

Object code only: US\$ 350; with source: US\$ 720. H & S included. No taxes. No royalties. Send check or money order, or ask for free info pack, including public domain MITUI Editor with source code.

◆ Request 104 on Reader Service Card ◆

SEEING IS BELIEVING WITH

DocBROWSER

DocBROWSER is a complete documentation system for C and PL/M programs, with printed reports on size, callers, users, declarations and more plus hot-key access to a full featured browser interface for all source and include files, functions, data structures and comments.

If you maintain programs, become an expert fast with DocBROWSER. Don't struggle through reams of listings when DocBROWSER gives you direct access to any part of your source with a single keystroke.



◆ Request 371 on Reader Service Card ◆

Performance Protocols

LAT™ XNS Telnet rlogin

Portable C Libraries
UNIX, VMS, DOS Ports

Fast Implementation
Fast Execution Speed
Fast Time to Market

Call 314.532.7708

Meridian

TECHNOLOGY CORPORATION
11 McBride Corporate Center Drive
Chesterfield, Missouri 63005-1406

Source code CD-ROM

Over 500 megabytes of source code archives. Thousands of programs, full source code for utilities, programming tools, games, etc.

\$39.95

PDQ Software
1547 Palos Verdes Mall
Suite 260
Walnut Creek, CA 94596

1-800-786-9907
1-510-947-5996

◆ Request 100 on Reader Service Card ◆

2) omitted the bookkeeping: how does TURBO C access this new library?.

Comment 1: (Probably brought on by some 30 years of teaching) Why not keep it simple? Give the questioner a simple "Hello World" example that stresses library use by itself?

Comment 2: This second comment is much more important and is the result of my telephone call to Borland: Where does one put his library so that Turbo C can find the functions contained in it? The ease with which one can write and save header files (in the same directory as other header files are saved in) would imply that personal libraries could be saved in the library directory. This is not the case. Turbo C is "hard wired" to search only the Borland-supplied libraries. Path(s) to users' libraries must be supplied in make, or project files.

(TLIB does allow you to add functions to (or modify) a current Borland library, but this could prove a dangerous thing to do.)

Thus, when using the very convenient IDE, every program that uses a non-Borland library must be a project, and a project file must be written!

Access to your own library is not explained where it should be (not in the Borland manuals, nor in the two classic Turbo C Bibles by Barkakati), nor can I find it in any of the many Turbo C books that I have compulsively bought. Only H. Schildt, in his *Turbo C, The Complete Reference*, Borland-Osborn/McGraw-Hill, gives this little gem of information. Even here, a confusing (and extraneous?) "T" is introduced.

Indeed, this omission on the part of many authors who glibly write of generating one's own library makes me wonder how many have ever done so?

Owen Gailar

Professor Emeritus, (Purdue University)
345 W. San Carlos
Fresno, CA 93704

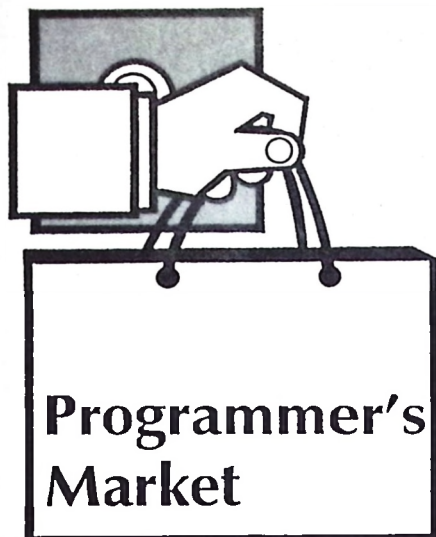
I have, and I had to learn all the things you and Ken summarized here. Thanks for the clarifications. — pjp

Dear Mr. Plauger,

I would like to respond to two items in the December *CUI*.

First, with regard to C. Skelly's fascinating article, "Creating C++-Like Objects in C." I believe Mr. Skelly made two important points that are worth reiterating.

One, callback functions are the foundation of polymorphic behavior. Understanding this point makes it easier to design flexible systems in C. This point also clarifies a common misconception about C++. The *virtual* keyword is



32-bit Protected Mode 386 C Graphics Library

Intel 386/486 C Code Builder
MetaWare, MicroWay, SVS,
Watcom & Zortech with
Phar Lap 386|ASM
Mixed Raster/Vector,
Scalable, Rotatable Font,
VGA, SVGA, 8514/A, VESA,
Hercules Graphics Station
through 1024x768x256 (8-bit),
640x480x32k (16-bit),
512x480x16.7m (32-bit),
WYSIWYG HP-GL/PostScript
\$200 NO ROYALTIES
FULL SOURCE CODE
Gary R. Olhoeft

P.O. Box 10870 Edgemont
Golden, CO 80401-0620
303-877-3697 CIS 76665,2021

◆ Request 475 on Reader Service Card ◆

OPT-TECH SORT/MERGE

Extremely fast Sort / Merge /
Select utility. Run as an MS-
DOS command or CALL as a
subroutine.

Supports most languages and
filetypes including Btrieve and
dBase. Unlimited filesizes, mul-
tiple keys and much more!

MS-DOS, Windows \$149.

OS/2, UNIX \$249.

Opt-Tech Data Processing

P. O. Box 678
Zephyr Cove, NV 89448
(702) 588-3737

◆ Request 199 on Reader Service Card ◆

Development Utilities

World's largest and best collections of
PD/Shareware for PC pro extensively indexed and
ZIPed for best value. 30 day guarantee.
Visa/MC/AmEx/COD. Ship/H \$5US, \$20Foreign.

Products	Disks/Files	Price
1-2-3 and compatibles	23/250	\$59.50
Assembler	23/300	\$59.50
C (Turbo & MS)	92/610	\$99.50
C++ (subset of above)	21/129	\$59.50
AutoCAD	17/570	\$59.50
dBase & Compilers	122/2000	\$149.00
DOS (PC consultants)	37/330	\$59.50
DTP (esp. Ventura)	70/400	\$79.50
Netware	67/504	\$99.50
Paradox	12/101	\$59.50
PC Products Database	41,000 records	\$25.00
Turbo Pascal	49/500	\$79.50
Visual BASIC	18/170	\$59.50
Windows	119/716	\$149.00
WordPerfect	39/290	\$59.50

EMS Professional Shareware
4505 Buckhurst Ct.; Olney, MD 20832
(301) 924-3594, Fax: (301) 963-2708

◆ Request 119 on Reader Service Card ◆

= The Window BOSS =

The Window BOSS is a Windowing and
Data Entry library for "C" compilers from
Microsoft, Borland, Watcom, Zortech,
Lattice, and MIX. It is small fast, power-
ful, and uncluttered. The programmer
interface is flexible and easy to recall.
Computer Language Magazine (9/91)
ranked The BOSS as a low cost high end
product with the best form-construction
capabilities of all the products reviewed!!
PC Resource, PCM, and other magazines
have printed similar accolades. Best of all,
The Window BOSS is shareware! Order
your shareware copy from the user group
or from Star Guidance. Order source
code from Star Guidance.

Star Guidance Consulting, Inc. - Waterbury, CT.
(203) 574-2449
Shareware: \$20.00 MC/VISA/COD Source: \$55.00

◆ Request 163 on Reader Service Card ◆

C Source Code

gnu c++ - includes c.....	\$135.00
gnu emacs.....	\$120.00
gnu c - 386 exe & DOS extender	\$115.00
gnu c - source only	\$90.00
pc/ip - tcp/ip, NFS & many drivers	\$50.00
gnu gdb - symbolic debugger	\$45.00
gnu gawk - awk clone	\$30.00
gnu ghostscript - PS clone.....	\$25.00
gnu gas - as for 80386, 68K, etc.	\$25.00
gnu binutils - ld, ar, nm, etc.	\$25.00
gnu bison and flex	\$25.00
gnu tools - grep, tar, diff, & rcs/cvs...	\$25.00
gnu fileutils - ls, mv, cp, etc.	\$25.00
PCcurses + libs - Turbo & MSC....	\$25.00
gnu games - Chess, NetHack, Go.....	\$25.00

Netkits - GNU for DOS

RCS - Revision Control System.....	\$50.00
Diff and Grep.....	\$35.00

DOS or CPIO
VISA/Master Card
Add \$4 for COD
US Postage Paid
Add \$10 for Next Day
Foreign Add 5%

NETWORKS
23 Cornwell Road
Freehold, NJ 07728
(908) 206-0320 Voice
(908) 303-9364 FAX

◆ Request 464 on Reader Service Card ◆

Learn C

Our C tutorial uses a unique method to teach you to program in this modern language. There are about 80 source files, on disk, which you study, then compile and execute to learn all aspects of ANSI-C.

Learn C++

Once you know C, you can go on to learn the newer constructs of C++. Our C++ tutorial teaches you "why" you should use object-oriented programming in addition to "how" to use it.

Each MS-DOS tutorial is \$39.95 + \$3.00 P&H (in US). M/C or Visa accepted. Other language tutorials are available also.

Coronado Enterprises

12501 Coronado Ave. NE
Albuquerque, NM 87122
(505) 293-5464

◆ Request 273 on Reader Service Card ◆

Hire a Marketing Pro for \$87 a year!



Get practical answers to your tough marketing questions.

The *Inside Trac*, a bimonthly interactive newsletter from The Trachtman Group, Inc. helps software companies navigate through the maze of marketing issues confronting product launches.

- Learn how to find your marketing program with initial direct and OEM sales.
- Find out how you can intelligently set the price of your new product.
- Discover how to get noticed in an increasingly crowded market.

The *Inside Trac* includes membership to an interactive bulletin board where you can share ideas with others trying to launch their products and get answers to your specific questions.

CALL 800-659-7420 for your first no-obligation issue FREE

The Trachtman Group, Inc.

Successful Marketing of PC Software for over 10 years



◆ Request 125 on Reader Service Card ◆

Why you want BATCOM!

BATCOM is a batch file compiler that compiles your ".bat" files to ".exe" files to make them faster, more professional, and more capable. BATCOM extends DOS with new commands so you can read keyboard input, perform arithmetic, use subroutines, and much more all from within compiled batch files. In addition to speeding your batch files and adding new commands, BATCOM protects your source code, and you can distribute your compiled programs with no royalties. For IBM PC. \$59.95.

Wenham Software Co.
5 Burley St.
Wenham, Ma. 01084
(508)-774-7036

◆ Request 138 on Reader Service Card ◆

WindowsCreator revolution!!!

- create WINDOWS interactively
- no need to write any code
- You PAINT the application, not write!!!
- object-oriented programming using standard C language
- Generated C-code or Smalltalk code in seconds!!!
- Dialog, window, menu, cursor, icon, color, brush and attrib. editors incl.
- interactive functionality linking
- application code is separated from interface code. Few code lines connect application to WINDOWS interface

AdamSoft, 66 rue de Bourgogne
L-1272 Luxembourg

Fax: +352-494768 VISA accepted
Until April 30th — \$200, after \$299
p&p Europe \$10, elsewhere \$20

◆ Request 146 on Reader Service Card ◆

C++ Multitasking Class Library

Now you can inherit the power of OBX™ Task Class

Serious developers REUSE to get a multitasking prototype fast!

See how easy multitasking can be: call for our ShellVision™ source code demo, you'll understand.

Supports Borland. Source available.

C::napse

: obx

11 Laurier, Suite 102,
Chambly, QC, Canada J3L 5S3
(514) 447-7221 Fax: (514) 447-7297

Z-PHIGS™

Use the power of most advanced graphics development system available today.

Z-PHIGS is an enhanced implementation of the only recognized 3D-graphics standard PHIGS+ for DOS and Windows. It provides everything needed to develop high-performance graphics applications. A powerful library with 2D/3D functions and a highly optimized data management system save you years of development time in the fields of CAD/CAM, Simulation, Multimedia, etc. Most sophisticated rendering capabilities like Lighting, Shading, Reflections, Texture-Mapping, etc. makes it simple and quick to give your Windows products those spectacular 3D-graphics features that will put your competitors to shame. Language bindings are available for Pascal or C.

For DOS \$795, for Windows \$1590
VISA and MasterCard accepted.

WISE Software, Seelandstr. 3, 2400 Lübeck 14, Germany
Tel: (+49) 451-3909-413 Fax: (+49) 451-3909-499

◆ Request 364 on Reader Service Card ◆

BGI GRAPHICS PC TIMER TOOLS

BGI Printer Driver Toolkit

Now you can effortlessly add high resolution hardcopy to your BGI graphics routines with BGI drivers for Epson, Proprinter, LaserJet, DeskJet, PaintJet, Postscript, and others. Supports TC, TC++, BC++, TP. Complete driver source code included. \$89.95

BGI For Windows

Port your BGI graphics code to Windows with our new BGI compatible graphics interface for Windows 3.x. Each window is a complete BGI environment with stroke font and hardcopy support. Supports BC++, TCW, TPW. Full source code included. \$69.95

PC Timer Tools

High resolution timing, delay, profiling, timer tick mgt., and scheduling functions. Supports TC, TC++, BC++, MSC, TP, Intel 386 Code Builder, and Zortech C++. \$69.95 with source.

PC Timer Objects

OOP version of PC Timer Tools for TC++, BC++, TP, and Zortech C++. \$69.95 with source.

Add \$4 each shipping USA, \$7 each elsewhere. VISA & MC are welcome. Our 30 day "No Questions Asked" return policy assures your satisfaction.

Ryle Design *Purveyors of Big Science*
PO Box 22, Mt. Pleasant Michigan 48804
Voice/Fax: 517.773.0587 BBS: 517.772.2393

◆ Request 110 on Reader Service Card ◆

PostScript TSRs — Under 9K

PSFX TSR prints PostScript output, just as formatted for Epson or IBM ProPrinter. See graphics, fonts, accents and boxes! Choose portrait or landscape, sizes up to 11x17. Print one or two pages per sheet. Use with databases, accounting, PrtSc key. \$85

PSFXnet Novell NetWare version adds banner pages. Keeps statistics on pages printed by each userID. \$99

EPScreen Captures PC screens as tiny Encapsulated PostScript (EPS) files (3K), frames & TIFF optional. Illustrations for manuals. Font included. Color files less than 10 K. \$95

Legend Communications, Inc.
54 Rosedale Avenue West
Brampton, ON, Canada L6X 1K1
30 day guarantee

(800)668-7077 (416)450-1010

◆ Request 192 on Reader Service Card ◆

'C' and 'C++' DOCUMENTATION TOOLS

! AUTOMATED DOCUMENTATION !

- C-CALL (\$59) Creates a graphic-tree of the caller/called functions, and creates a files-vs-functions table of contents.
- C-CMT (\$59) Creates/inserts/updates comment-blocks for each function, listing the functions and identifiers used by it.
- C-METRIC (\$49) new! Calculates the path 'cyclomatic' complexity of functions, counts lines of Comments, Code, 'C' statements.
- C-LIST (\$49) Lists and action-diagrams, or reformats source into standard formats.
- C-REF (\$49) Creates cross-reference of local/global/define/parameter identifiers.
- SPECIAL OFFER (\$189) All 5 programs, fully integrated as C-DOC DOS program, plus free OS/2 protected-mode program.
- 30-DAY Money-back guarantee CALL NOW

SOFTWARE BLACKSMITHS INC.

6064 St Ives Way, Mississauga
ONT, Canada, L5N 4M1 (416)-858-4466

See AD INDEX for our larger ad

◆ Request 173 on Reader Service Card ◆

not free. There is an overhead associated with using virtual functions, which is exactly the overhead of the call by address mechanism.

Two well-constructed C libraries use subject-predicate naming conventions, instead of the more natural seeming predicate-subject. For example, *Circle-Draw()* is correct, *DrawCircle()* is incorrect. Having our function names sort on the object name *Circle* allows us to associate the functionality of the *Circle* structure with those functions that begin with that name. This system also has the nice property of sorting well in the symbol tables.

The second point I wish to address refers to Jack Purdum's letter on the *define* versus *declare* problem. I am surprised you don't get it. Stating a thing twice doubles the amount of work needed to maintain it. Any symbol addition in Mr. Purdum's example requires one source code modification, as opposed to the usual two. This problem occurs in a lot of places. Preprocessor tricks, which are all variations on the one given, can save days of hard boring labor in the course of a large project.

Sincerely,

Adam Greissman, President
Hermeios, Inc.
853 Broadway, Suite 1104
New York, NY 10003

I see both advantages and disadvantages in SubjectPredicate naming. I see both advantages and disadvantages in macros that let you declare a data object just once. That makes it hard for me to characterize any of these approaches as "right" or "wrong." — pjp

Dear Mr. Plauger:

I would like to request an article discussing the NCEG extensions to Standard C be published. In addition, perhaps several articles discussing issues regarding their use.

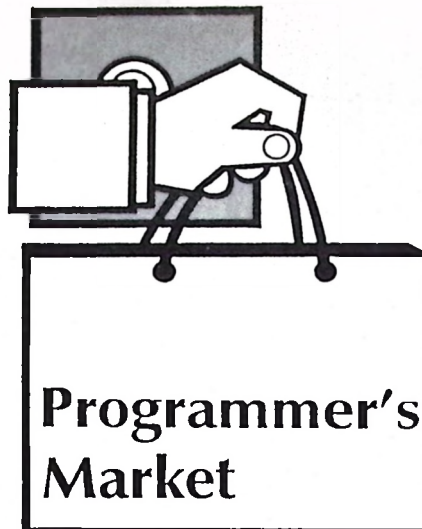
As an avid reader of your column, I would like to know your opinions regarding text formatting packages, especially regarding *nroff/troff* versus the current crop of desktop publishing packages. Would you care to clarify it in an upcoming column?

Thanks for your help.

Sincerely,

Wen J. Chen
143-25 41st Avenue, #340
Flushing, NY 11355

I haven't discussed NCEG issues in my column yet for two reasons. First, I still have a few more installments on the Standard C library. Second, the NCEG folk are still settling down on the content of their technical report. I fully expect to devote some space to their work at the appropriate time. — pjp



TURN A PC INTO A POWERFUL SERIAL COMMUNICATIONS PROTOCOL ANALYZER

COM LAB VERSION 1.01 \$395.00
VERSION 2.0 \$790.00

THE FEATURES & PERFORMANCE OF AN EXPENSIVE HARDWARE ANALYZER AT A FRACTION OF THE COST... PLUS YOU CAN CREATE CUSTOM CAPTURE & ANALYSIS PROGRAMS WITH OUR BUILT IN EDITOR & "C" COMPILER

- Use standard COM1 / COM2
- NO special cables required
- usec timing accuracy
- Easy to use menus & user defined windows
- Data To & From disk
- Capture & Transmit in both directions
- SYNC or ASYNC

REALTIME CONTROL, INC
(904) 373-2626
(800) 232-0485

◆ Request 388 on Reader Service Card ◆

as you've never seen it before with

Stylus

- Read existing C source code into structure chart form (machine readable NS)
- Create and/or maintain code at the Nassi Shneiderman chart level (no more braces!)
- Automatically translate NS charts back into conventional C source for compilation
- Runs under DOS and OS/2 full screen mode with extensive help facilities
- \$295.00 includes 1st class shipping. Checks, Visa, American Express and Mastercard accepted.

Stylus Software
144 Two Appletree Square
Minneapolis MN 55425-1637
Tel: (612)-854-5800 FAX: (612)-854-6948
Toll Free: 1-800-245-2575

◆ Request 196 on Reader Service Card ◆

FLOATING POINT ARRAY PROCESSOR

- ★ Operates on Arrays & Matrices; Integer, Real, & Complex Structures.
- ★ Process math-intensive algorithms 100 times faster than the 80387.
- ★ 598 mathematic & scientific functions on-board in EPROM, callable from C.
- ★ Private high speed STATIC memory (0.25 to 4.25 megabytes) on board.
- ★ 3 megabytes/sec to/from ISA host.
- ★ Direct hardware link to video & A/Ds.
- ★ 1000 page reference manual.
- ★ \$2495.00 complete with software.
- ★ Call for function list and benchmarks.

Eighteen Eight Laboratories

1-800-888-1119 FAX 702-294-2611

1247 Tamarisk Lane
Boulder City, NV 89005

◆ Request 121 on Reader Service Card ◆

Installation Made Easy

The Stork Automatic Installation Utility

Easy to use WYSIWYG design system walks you through each step of the installation process. Using mouse or keyboard, you create a customized installation procedure as you interact with each design tool. Extensive built-in error checking in the installation procedure as well as in the design system gives you confidence that the procedure is accurate and bullet-proof when used by your customer.

"This product has proved invaluable for both simple and complex installation procedures."

The Stork Delivers \$175 (No risk 30 day guarantee)

Island Systems

7 Mountain Rd, Burlington MA 01803
(617)273-0421 Fax (617)270-4437
MasterCard & VISA accepted

◆ Request 144 on Reader Service Card ◆

DEVELOPERS! ΔΕΞΑΟΠΕΙ!

GOING INTERNATIONAL?

Our string externalization utilities and C functions can help you get your package ready for the international market.

These proven utilities extract literal strings from your C source code, ready for editing, translation or encryption, without impacting your original executable. While saving scarce initialized data space, these functions allow your easily re-defined prompts to behave as if they were compiled directly into your program. This permits the original executable to "speak" the language of your choice, even Japanese!

Includes all object/source code and libraries for popular C compilers. Price \$149.95 for object, \$249.95 for source code site license. No royalties. DOS, Unix, Mac, etc. VA residents add sales tax.

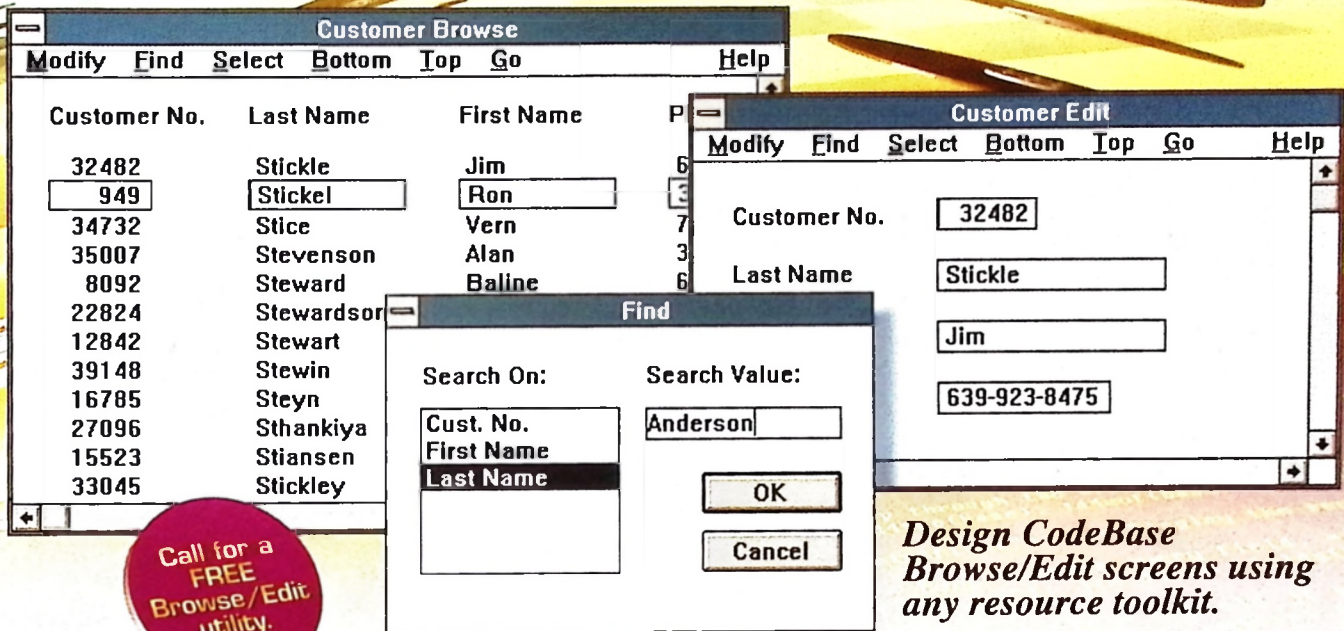
Network Dynamics, Inc.

2225 S. Henry Street, Suite L-2
Williamsburg, VA 23185

Phone (804) 220-8771 Fax (804) 220-5741

◆ Request 419 on Reader Service Card ◆

Access dBASE IV and FoxPro files from C or C++



- Multi User
- Portable (DOS, Unix, ...)
- Royalty Free DLL
- C++ interface included

Use CodeBase 4.5 from Visual Basic or Turbo Pascal for Windows.

Use the super-fast, super-small FoxPro 2.0 CDX or the Clipper NTX index files.

" Our product was too slow under FoxPro 2.0, so we rewrote it in C using CodeBase. Now it is incredibly fast."

Jeff Reed, DCS Computer Services



CodeBase 4.5

The C Library for DataBase Management

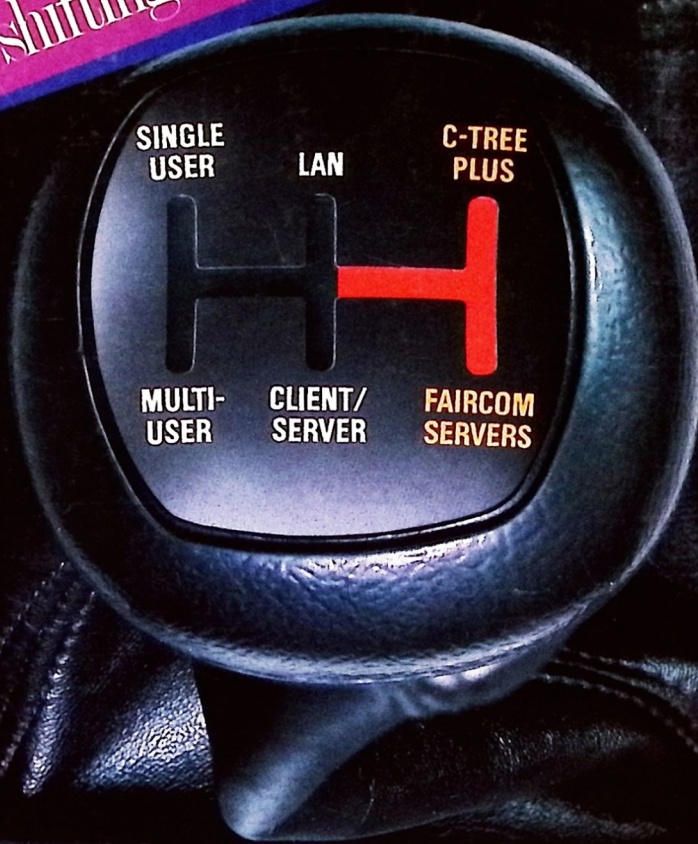
SEQUITER
SOFTWARE INC.



TEL. 403-437-2410
FAX 403-436-2999
Europe 33.20.24.20.14

#209,9644-54 AVE., EDMONTON, AB, CANADA T6E-5V1

Tired of shifting development gears?



Let c-tree Plus™ and the FairCom Servers put you into overdrive.

■ Use c-tree Plus as a data management engine, or as a front-end to the FairCom Server or FairCom SQL Server!

Now you don't have to choose between different application configurations — c-tree Plus supports them all! You can use c-tree Plus to create single user, multi-user, or LAN applications, and easily adapt them to utilize the advanced client/server technology of FairCom Server or FairCom SQL Server, (which offers full ANSI-standard SQL functionality).

■ c-tree Plus offers complete scalability!

No matter what kind of platform you're running on (or moving to), c-tree Plus eliminates the tedious, time-consuming reprogramming normally required to move from one environment to another. With c-tree Plus, you don't change your source code at all — just recompile, link, and you're running on platforms ranging from Cray supercomputers to Zenith laptops!

■ With the FairCom Servers, c-tree Plus offers powerful (and affordable) transaction processing capabilities!

Using c-tree Plus as a front-end to the FairCom Servers, you can incorporate powerful transaction processing features directly into your applications, including full commit/rollback, intermediate savepoints and complete logging.

■ Order c-tree Plus, and put yourself in the driver's seat!

Whether you currently need the power of client/server technology, or just need the most powerful data management system around, c-tree Plus is for you — order your copy today!



FairCom Servers can be tightly bound to your application.

c-tree™ Plus

**FairCom
Server**

◆ Request 128 on Reader Service Card ◆



FAIRCOM
corporation

4006 WEST BROADWAY
COLUMBIA, MO 65203
PHONE 314 445 6833
FAX 314 445 9698

(800) 234-8180, Ext. 1